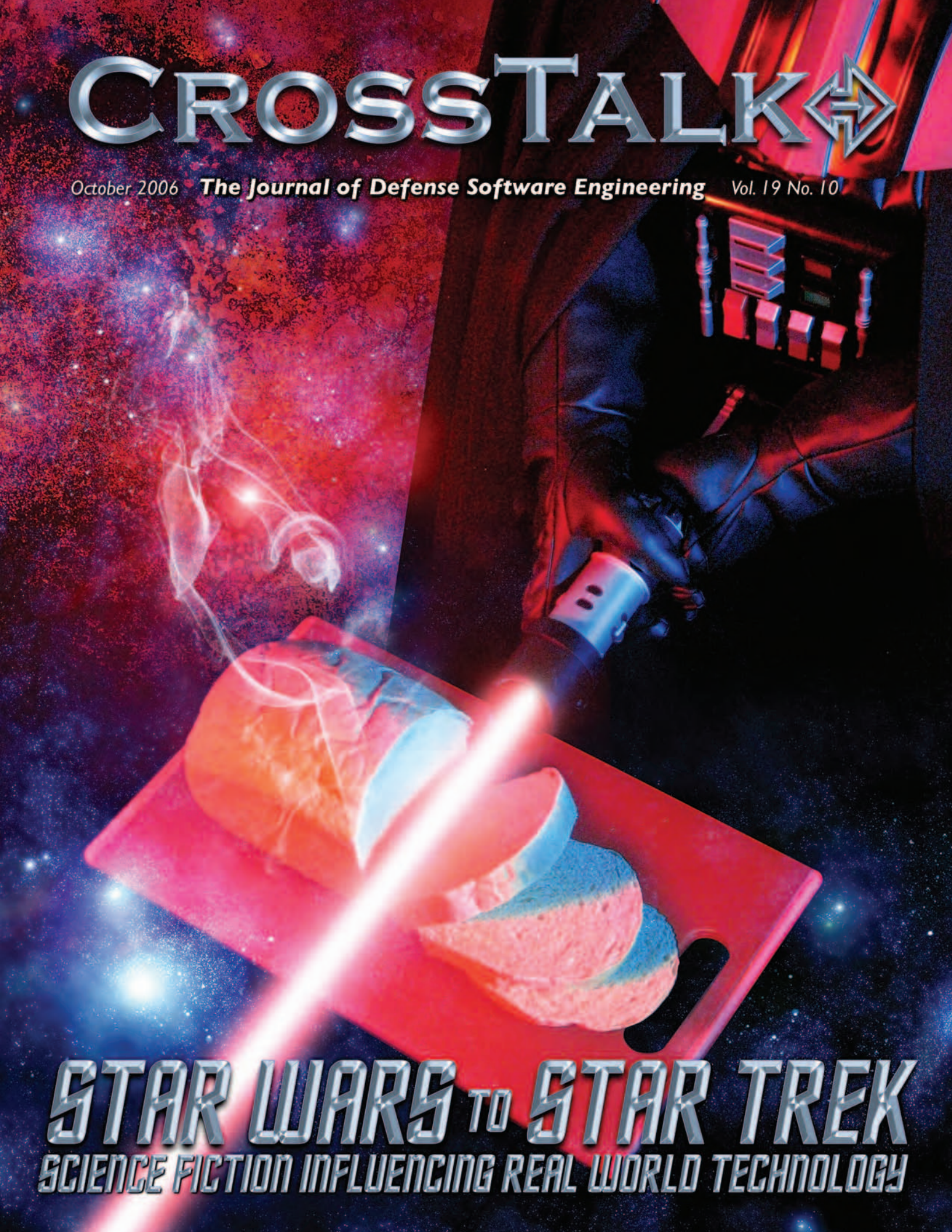

CROSSTALK
October 2006 The Journal of Defense Software Engineering Vol. 19 No. 10
STAR WARS TO STAR TREK
SCIENCE FICTION INFLUENCING REAL WORLD TECHNOLOGY

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
OCT 2006		**00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
CrossTalk: The Journal of Defense Software Engineering. Volume 19, Number 10, October 2006	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **OO-ALC/MASE,6022 Fir Ave,Hill AFB,UT,84056-5820**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**32**	

Star Wars to Star Trek

Software Engineering Technology

Open Forum

On The Cover
Cover Design by Kent Bingham

Additional art services provided by Janna Jensen

Departments

CrossTalk

76 SMXG Co-Sponsor	*Kevin Stamey*
309 SMXG Co-Sponsor	*Randy Hill*
402 SMXG Co-Sponsor	*Diane Suchan*
DHS Co-Sponsor	*Joe Jarzombek*
NAVAIR Co-Sponsor	*Jeff Schwalb*
Publisher	*Brent Baxter*
Associate Publisher	*Elizabeth Starrett*
Managing Editor	*Kase Johnstun*
Associate Editor	*Chelene Fortier-Lozancich*
Article Coordinator	*Nicole Kentta*
Phone	(801) 775-5555
E-mail	crosstalk.staff@hill.af.mil
CrossTalk Online	www.stsc.hill.af.mil/crosstalk

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the U.S. Air Force (USAF), the U.S. Department of Homeland Security (DHS), and the U.S. Navy (USN). USAF co-sponsors: Oklahoma City-Air Logistics Center (ALC) 76 Software Maintenance Group (SMXG), Ogden-ALC 309 SMXG, and Warner Robins-ALC 402 SMXG. DHS co-sponsor: National Cyber Security Division of the Office of Infrastructure Protection. USN co-sponsor: Naval Air Systems Command.

The USAF Software Technology Support Center (STSC) is the publisher of CrossTalk, providing both editorial oversight and technical review of the journal. CrossTalk's mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

S T S C

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail us or use the form on p. 20.

517 SMXS/MDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CrossTalk does not pay for submissions. Articles published in CrossTalk remain the property of the authors and may be submitted to other publications.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: This Department of Defense (DoD) journal is an authorized publication for members of the DoD. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, or the STSC. All product names referenced in this issue are trademarks of their companies.

Coming Events: Please submit conferences, seminars, symposiums, etc. that are of interest to our readers at least 90 days before registration. Mail or e-mail announcements to us.

CrossTalk Online Services: See <www.stsc.hill.af.mil/crosstalk>, call (801) 777-0857 or e-mail <stsc.webmaster@hill.af.mil>.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

The Vision of What Can Be

This month's CROSSTALK theme evokes references to *Star Wars* and *Star Trek* to remind us of the influence of science fiction on real world technology. For me, the theme triggered a trip down memory lane.

When *Star Trek* debuted in 1966, I was a freshman in college. Computers were huge mainframes housed in data processing centers, fed by 80 column cards which you were warned not to bend, fold, mutilate, or spindle or scan sheets which required a No. 2 pencil. Pocket calculators still had limited functionality and were beyond my price range. By the time *Star Wars* premiered in 1977, I was the proud owner of a Texas Instrument calculator which had about eight functions, an ATM card which allowed me to withdraw up to $60 cash if the computer wasn't down, and I had heard of the Apple computer but didn't know anyone who owned one.

So where are we today? And did science fiction get us here? Clearly real world technology has improved exponentially over the last 40 years. Unmanned combat air vehicles, precision guided weapons, global positioning system navigation, and digitization of the battlefield have transformed our warfighting doctrine. Thumb drives, cell phones, pagers, text messages, digital cameras, and wireless internet access are available to all and are affordable, enabling global communication and commerce.

Were these technological advances influenced by science fiction or merely inspired by it? The tagline for *Star Trek* to *boldly go where no one has gone before* provides the key. The technology advances of the past 40 years could not have happened if we chose to maintain the status quo. Transporter rooms, phasers set to stun or kill, and the noninvasive medical instruments used by Doctor McCoy may not be in everyday use, but they are part of our vision of what can be.

Diane E. Suchan
Warner Robins Air Logistics Center Co-Sponsor

Star Wars, Star Trek, and CROSSTALK

I love science fiction; *Star Wars* and *Star Trek* are two of my favorite themes. As I was studying electrical engineering (EE) in college, a friend pointed out that you cannot spell *geek* without EE. Later, I was listening to the results of a study of geeks on the radio. I tried to deny fitting the profile they presented, but I could not deny our favorite show was *Star Trek*. As a result, I truly looked forward to this issue, and enjoyed reading the articles when they were submitted for consideration.

Our first article, *The Science in Science Fiction's Artificial Men*, by Dr. Dawn MacIsaac and Dr. Kevin Englehart takes the reader from science fiction to the real-world applications, then ties these concepts into the needs of our soldiers. *A Brighter Future From Gallium Nitride Nanowires* by Dr. Kris A. Bertness, Dr. Norman A. Sanford, and Dr. Albert V. Davydov discusses how nanowires are replacing current technology. Our last theme article turns from technical issues to managerial issues in *Leadership, The Final Frontier: Lessons From the Captains of Star Trek* by Paul Kimmerly and David Webb. While too light-hearted for our theme section, the article *All We Need to Know About Software Project Management, We Can Learn From Watching Star Trek* by David Webb provides some real insights worth considering during software development.

I truly enjoyed putting this issue together partly because of my love of science fiction and partly because of the articles we received. Whether you are a true geek or just enjoy new technology, I hope you will enjoy this issue and the insights within its pages.

Elizabeth Starrett
Associate Publisher

The *Science* in Science Fiction's Artificial Men

Dr. Dawn MacIsaac
Institute of Biomedical Engineering

Dr. Kevin B. Englehart
University of New Brunswick

Bioengineers are working around the clock to meet science fiction's standards for electronically powered artificial limbs. Combining state-of-the-art embedded systems and real-time software solutions with some innovative bio-interfacing strategies may just meet the challenge. Science fiction has plenty of fantastic examples of humans made from artificial parts: Star Trek*'s Lieutenant Commander Data and* Star Wars*' villain Darth Vader stand out among them. Data, controlled by his positronic brain, is a full-fledged android. Vader, on the other hand, remains some part human, although the ominous mask he wears is a constant reminder of the artificial components he must don to stay alive. From the perspective of the storytellers, we have mastered the science of mimicking humans with technology. Such artificial humans are effortlessly woven into these fictional storylines which leave reality with the daunting, complex task of actually engineering such achievements. With a quick trip to the lab, Luke Skywalker is fitted with a fully functional artificial hand to replace the one he lost in a battle against Darth Vader. Compared to Vader or Data the android, Skywalker's hand seems almost trivial ... in fiction maybe, but reality tells a different story.*

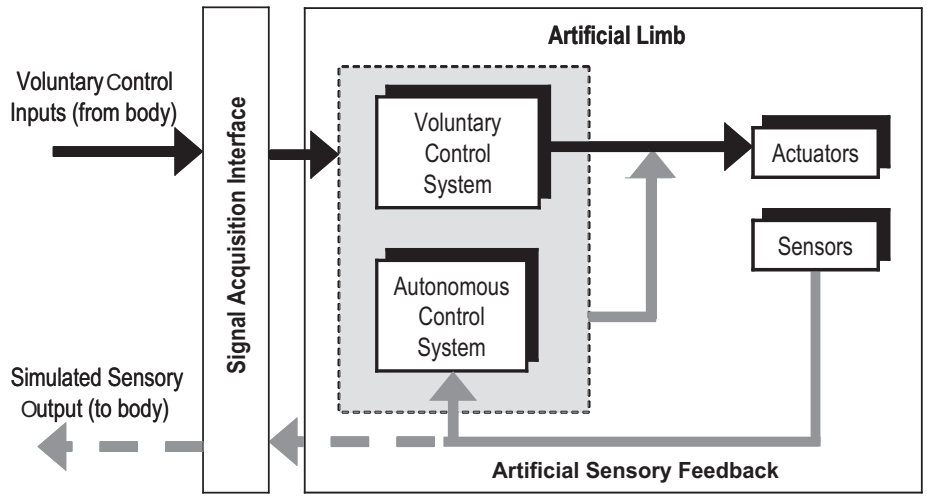
Although Luke Skywalker's hand seems trivial compared to the technology used for Lieutenant Commander Data and Darth Vader, compared to modern technology, it emulates our current efforts to create a synthetic, symbiotic robotic arm.

Figure 1 depicts an overview of the components necessary to make up an artificial limb such as the one used to replace Skywalker's hand. At the heart of this system are the voluntary and autonomous control components, generally encapsulated in a coordinated digital signal processing (DSP) microcontroller and capable of real-time processing. These two components control the functional branches of the system as shown by the black and gray arrows.

The first branch, indicated in black, allows the user control over basic motions required of the limb, for example, grabbing an object. The second branch, indicated in gray, allows the limb to autonomously fine-tune its activities based on sensory feedback made available to the system through sensors within the limb itself. This functionality is more subtle than voluntary control because it usually goes unnoticed when we do it with a natural limb. For instance, when we grab an egg, we do not think about how hard we should grasp. Instead, our nervous system automatically takes care of that for us so that we can grab the egg without breaking or dropping it. The dashed gray arrows indicate a sub-branch of this function, allowing users to be made aware of the sensory feedback provided by the hand. For instance, we can feel when our grasp is slipping in a natural hand, and a fully functional artificial hand should provide a similar function.

Modern developments in microcontroller, micromotor, and microsensor technologies yield promising avenues for continued development of the front-end actuation and fine-tuning efforts necessary for a fully functional artificial limb. The real challenge for bioengineers pioneering this technology is the back-end voluntary control and sensory perception function. Moreover, the challenge becomes even more daunting as the degrees of freedom for the artificial limb increase, yet the solution becomes more essential for the user.

Figure 1: *Overview of the Components That Make Up an Artificial Limb*



Conventional Control Systems

Myoelectric control has found widespread use as a voluntary control strategy in upper-limb powered prosthetics. Using this approach, voluntarily controlled parameters of electrical signals from muscles are used as inputs to modulate prosthesis function. These electrical signals, which can be measured noninvasively by a pair of electrodes placed on the surface of the skin, are called myoelectric signals.

Early myoelectric controllers operated in an on/off mode to control prosthetic function. For instance, when the controller detected a signal from one muscle, it opened a prosthetic hand; when it detected a signal from another muscle, it closed the hand.

This simplistic control scheme is easy to implement in either analog circuitry or digital software, requiring only an estimate of the mean absolute value of the signal to compare to an on/off threshold. While simple, this control scheme is substantially limited since any additional functionality is dependent on the availability of additional muscle sites for control inputs, and no provision is made for speed control. Furthermore, electrode

placement to pick up single muscle signals is challenging, and voluntary contraction of the single muscles is often non-intuitive and difficult for users to learn. Consider, for instance, using the biceps and triceps muscles in the upper arm to open and close a hand. Nevertheless, artificial hands with such limited grasping functionality were making significant clinical impact by the 1970s and well into the 1980s [1]. The Otto Bock 2-state system was a common example [2].

The evolution of microcontroller technology including DSP chips with real-time processing power offered new opportunities for advancing powered prosthetic development. As microcontroller technology evolved, more sophisticated control schemes based on complex pattern recognition became possible. Research into the nature of the myoelectric signal has demonstrated that a given muscle within a muscle group will contribute variably to the overall group's signal depending on the intended limb action [3]. The sum of the contribution of all muscles within a muscle group will, therefore, reflect intended action-dependent patterns. Because of the random nature of the myoelectric signal, these patterns are difficult to extract, but with the advent of DSP chips like Texas Instrument's C2000 – found in the Boston Elbow developed by Liberating Technologies Inc. [4] – software that reliably interprets these signals can be embedded into the artificial hand's control system. Different classification strategies are currently being investigated including statistical; syntactic; and, most recently, machine learning via the perception-based neural network [5].

Increased computational resources and advancements in algorithm development have afforded bioengineers opportunities to explore some rich feature sets for input to the pattern classification software. Early pattern classification-based control systems were limited to simple-to-calculate time domain signal statistics such as variance, zero-crossings, and waveform length to represent the myoelectric signal of interest [1]. Now, far more computationally complex feature sets are being investigated, including autocorrelation coefficients; spectral measures; time-series model parameters; and time-frequency coefficients based on wavelet and wavelet packet transforms and higher-order spectral analysis [1].

As an example, a pattern recognition based control system developed at the University of New Brunswick (UNB) was used to recognize 10 discrete movements

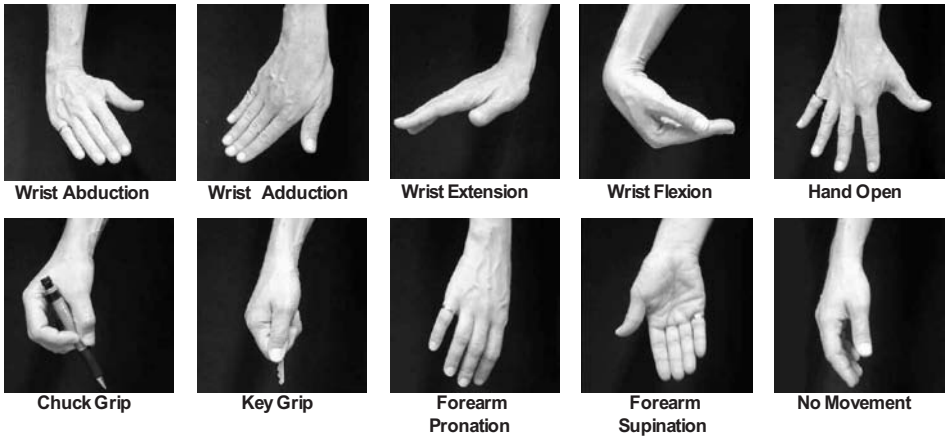


Figure 2: *Hand and Wrist Functions to be Restored in a Below-Elbow Amputee*

of the wrist and hand. This system would be used by an individual with an amputation below the elbow. The movements to be controlled are depicted in Figure 2.

Autoregressive coefficients and a linear discriminant classifier were used, and the system was trained for use by eight individuals. Sixteen electrodes were placed around the circumference of the forearm, as depicted in Figure 3. The performance of each subject was assessed by the percentage accuracy with which they were able to correctly select a randomly presented target movement. In Figure 4 (see page 6), the accuracy is shown for each subject with respect to the number of electrodes used[1]. Although performance varies between subjects, it is clear that the use of more electrodes yields better performance; no improvement exists beyond the use of eight electrodes. This system is remarkably accurate with an average user capable of selecting amongst these movements with an accuracy of 96 percent.

The pattern recognition strategy for voluntary control of powered prosthetics allows for more degrees of freedom than the simple, single-muscle/single-function control strategy because it can differentiate between many more intended limb actions. While this is an important step toward artificial limb technology which meets the standard of science fiction, real state-of-the-art systems are still limited. They are still dependent on the availability of muscle sites to elicit control signals, and they must control each joint in a serial manner; independent control of multiple joints is still an elusive task.

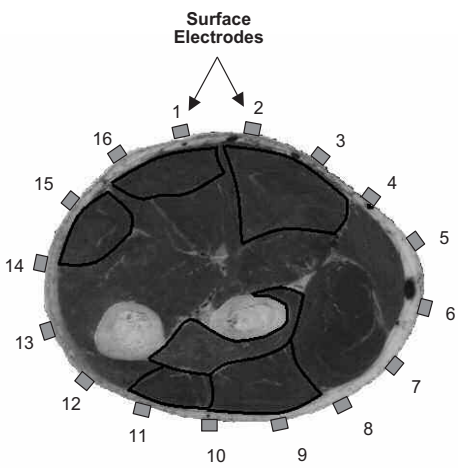
Emerging Strategies

Although steady progress has been made in myoelectric control systems, the use of the surface myoelectric signals in a conventional manner has inherent limitations. Individuals with high-level limb amputations (above elbow, for example) have few muscle sites from which control information may be derived, and they have more function which must be replaced. Consider an individual with an amputation at the shoulder. To restore lost function, the user must have a prosthesis capable of articulating many degrees of freedom, including the hand, wrist, elbow, and shoulder joints. The only myoelectric signals available for control are the pectoralis, some back muscles, and perhaps some remnants of the shoulder deltoids.

Therein lies a fundamental paradox: The higher the level of amputation, the more degrees of freedom must be replaced, with a diminishing number of control sites. Moreover, the available control sites are not physiologically appropriate; that is, the muscle activity bears no natural relationship with the lost degrees of freedom. Even the most sophisticated pattern recognition-based control system cannot defeat this paradox; only contrived contractions that are unrelated to the natural contraction patterns can be used to impart control.

The unfortunate consequence of this paradox is that those with high-level amputations are those in greatest need of

Figure 3: *The Placement of Surface Electrodes Around the Forearm*



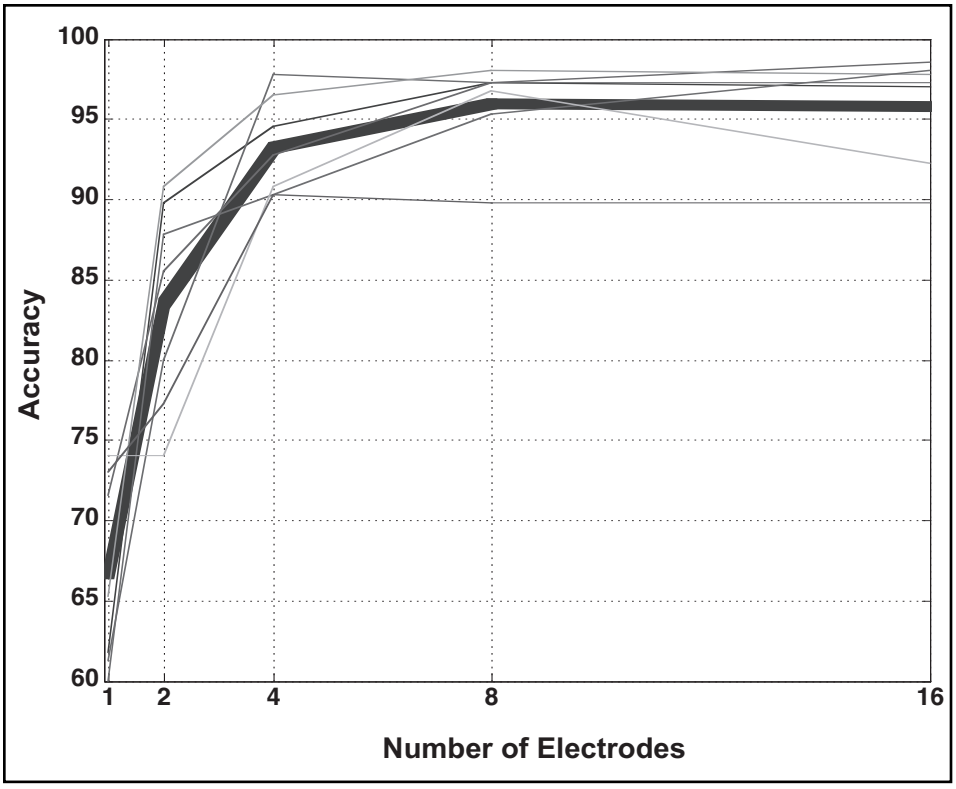


Figure 4: *The Accuracy of Each Subject in Selecting Between 10 Classes of Desired Motion. The Heavy Line Is the Average Performance Across All Subjects*

functional replacement. Individuals with a missing hand or wrist can perform activities of daily living quite well with or without a prosthesis. Those with amputation above the elbow or at the shoulder require substantially greater assistive augmentation.

The means of defeating this paradox lies in alternative sources of information. Clearly, the use of conventional myoelectric signals from residual muscle tissue cannot provide physiologically appropriate control sites. Fortunately, there are three emerging technologies that show great promise: Targeted Muscle Reinnervation (TMR), Peripheral Nerve Interfaces, and Cortical Interfaces.

TMR

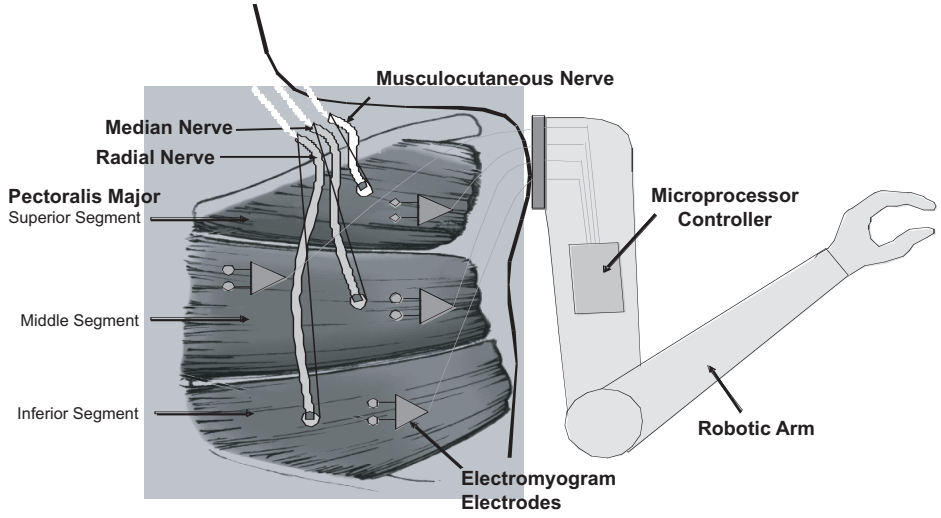
Often, residual nerves can remain intact after amputation and retain the capacity to transmit messages from the brain; they just do not have anywhere to transmit the information. TMR is a surgical procedure which transfers residual nerves from an amputated limb onto alternative muscle groups. The target muscles are not biomechanically functional since they are no longer attached to the missing arm. The reenervated muscle then serves as a biological amplifier of the amputated nerve motor commands which are intuitively coupled to the intended action. The muscle thus provides physiologically appropriate, surface myoelectric control signals that are related to functions in the lost arm and allow simultaneous control of multiple degrees of freedom in an advanced prosthesis.

TMR is an innovative strategy for interfacing neural commands from the brain with an artificial limb. The artificial limb must still be fitted with an embedded system capable of supporting the software required to interpret the complex patterns in the myoelectric signal provided by the alternative muscles, and it must contain the software required to provide some simultaneous control which the input signals request. The intuitive nature of this voluntary control strategy makes TMR an extremely appealing development.

The first person to receive TMR was a 54-year-old male who had suffered severe electrical burns working as a high-power lineman in May 2001. He required bilateral shoulder disarticulation amputations [6]. To improve the function of his powered left prosthesis, TMR surgery was performed in February 2002. The patient's pectoral muscles were denervated, divided into four separate segments, and a residual arm nerve was transferred to each segment (Figure 5).

Within five months after the surgery, surface myoelectric signals could be recorded from the pectoral segments, reenervated with the musculocutaneous, median, and radial nerves. In this subject, TMR allows the musculocutaneous nerve transfer to control elbow flexion, the radial nerve transfer to control elbow extension, the median nerve flexor region to control hand closing, and the median nerve thumb abductor region to control hand opening. Fitted with a sophisticated artificial arm that interfaced with the pectoral muscle group, the subject was able to operate his elbow, wrist and terminal device simultaneously with greater ease and speed.

Figure 5: *Schematic Description of Targeted Muscle Reinnervation Technique*[2]



Peripheral Nerve Interfaces

Peripheral nerves deliver control information to skeletal muscle. As compared to surface myoelectric signals, substantially more information about motor intent is available in peripheral nerves if it can be reliably measured. Unfortunately, nerve signals are much harder to measure, as they are embedded in the body and are surrounded by a muscle that creates an interference signal. Most artificial limb voluntary control systems have focused on myoelectric control inputs because no technology has been available to measure information from peripheral nerves. Recently, however, a great deal of research has been directed at measuring

peripheral nerve activity using cuff electrodes that envelop the nerve [7], biocompatible silicon sieves through which a severed nerve may regenerate and clamps which compress the nerve, exposing multiple fibers to a recording surface [8]. Perhaps the most encouraging approach is the use of a slanted electrode array, developed at the University of Utah [9]. The Utah slant array is a 100-electrode array with a resolution of approximately 400 microns between electrodes, as depicted in Figure 6.

Cortical Interfaces

Brain-Computer Interfaces (BCI) have been the subject of a great deal of research. An effective BCI would allow people with severe motor disorders such as paralysis, stroke, cerebral palsy, and spinal cord injury to control a device such as a robotic arm or a computer with signals recorded directly from their brain. Some promising advances have been made using surface electroencephalogram recordings [10]. Surface recordings require dozens of electrodes however and are therefore clearly not a practical approach for a system that a user must wear and maintain mobility.

Microelectrode arrays implanted in the motor cortex of monkeys have been shown to convey motor intent. Using this, it has been shown that it is possible to resolve cortical activity in a manner that describes limb trajectory and hand articulation [11]. Moreover, it has been shown that the pre-motor area of the brain can convey task-planning activities [12], which provides higher level information about motor intent. Although very promising, substantial biomedical challenges remain, such as implanting sensors in the appropriate locations, ensuring the biocompatibility, and powering these sensors for long periods of time.

Limb trajectory information from the motor cortex describes intent in Cartesian space (ordinary *two-* or *three-dimensional* space), with respect to the position of the hand. This is in contrast to the information from the myoelectric signal and the peripheral nerves, which impart control to the *joints* of the upper extremities. One may say that this information resides in joint space. A fusion of cortical information in Cartesian space and myoelectric signal/nerve signal information in joint space is likely the most robust approach to dexterous artificial limb control. With further developments in cortical interfaces, smart systems which integrate these information spaces may yield a fully functional, voluntarily controlled artificial limb.

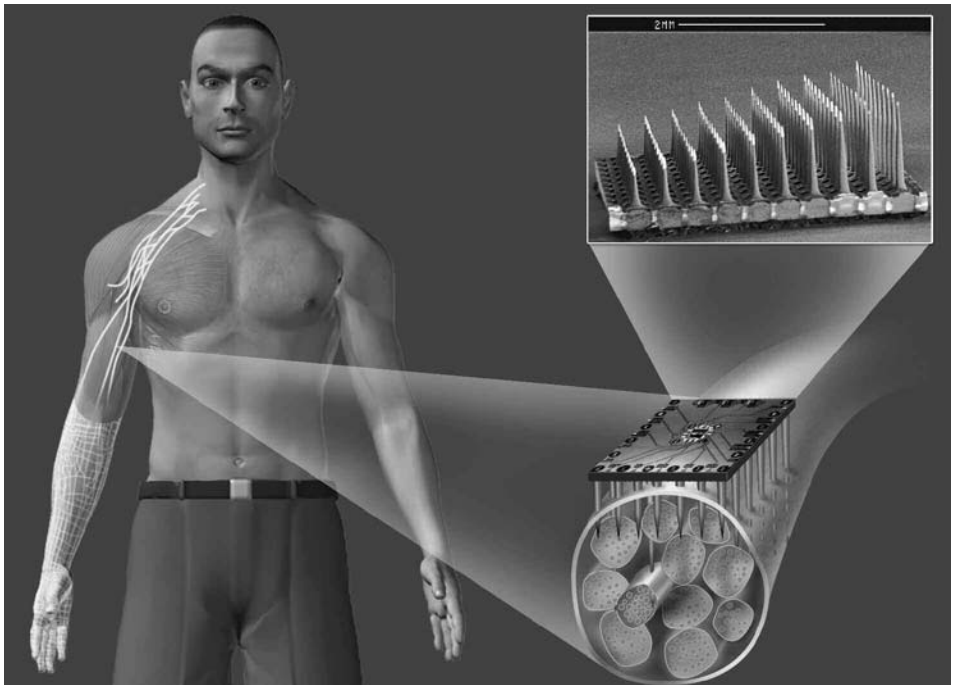


Figure 6: *The Use of the Utah Slanted Electrode Array in the Measurement of Peripheral Nerve Activity*[3]

With the promise of these new sensor technologies, new advancements in biosignal processing software will have to be made. New information extraction algorithms may have to be developed, along with new control strategies, and a viable means of sampling and wirelessly transmitting the high density stream of data to the front-end control system must be devised.

Pushing Forward

The U.S. Defense Advanced Research Projects Agency (DARPA) has sponsored two initiatives that address these emerging technologies described above, with the resolute goal of delivering next-generation prostheses to soldiers and civilians in the near term. A two-year project, entitled *Prosthesis 2007* has been awarded to Deka Research (Manchester, New Hampshire), which aims to dramatically improve state-of-the-art technology in upper-limb prosthetics. This will be accomplished by using existing technology or near-term innovation to produce a limb that will allow the user to simultaneously control his or her shoulder, elbow, wrist, and hand. A four-year project, entitled *Revolutionizing Prosthetics* has been awarded to the Johns Hopkins University Applied Physics Laboratory. This initiative will realize a prosthetic limb that has function identical to an intact human limb in terms of dexterity and sensory perception. In addition to meeting the challenges of deriving neural information from users for control, these projects will significantly advance prosthetics technology with regard to electromechanical design (energy storage, actuation, transmission) and human factors (improved socket design, osseointegration).

State-of-the-art, electrically powered artificial limbs are a long way off from meeting the standards set by science fiction. However, there is no doubt that we are now, more than ever, committed to meeting this challenge. Given the progress we have already made, new advancements in surrounding technologies, including embedded and real-time software systems and the resolve of the bio-engineers and medical practitioners working to meet the challenge, we are likely to see reality close in on fiction in the near future.◆

References

1. Parker, P.A., K. Englehart, and B. Hudgins. "The Control of Upper Limb Prostheses." Electromyography: Physiology, Engineering, and Noninvasive Applications. Eds. R. Merletti and P.A. Parker. Wiley-IEEE Press, 2004.
2. Scott, R.N., and P.A. Parker. "Myoelectric Prostheses: State of the Art." Journal of Medical Engineering and Technology 12 (1998): 143-151.
3. Hudgins, B.S., P.A. Parker, and R.N. Scott. "A New Strategy for Multi-function Myoelectric Control." IEEE Transactions on Biomedical Engineering 40.1 (1993): 82-94.
4. Murray, C. "Rewiring the Body." Design News Oct. 2005 <http://www.designnews.com/article/CA6275330.html>.

Coming Events

November 5-8
AYE 2006
Amplifying Your Effectiveness
Phoenix, AZ
www.ayeconference.com

November 6-10
ISSRE 2006
The 17th IEEE International Symposium on Software Reliability Engineering
Raleigh, N.C.
www.csc2.ncsu.edu/conferences/issre/cfp.php

November 6-10
International Testing Certification Week
Columbus, OH
www.testinginstitute.com/stpw/columbus06

November 13-15
The 10th IASTED International Conference on Software Engineering and Applications
Dallas, TX
www.iasted.org/conferences/2006/Dallas/sea.htm

November 13-16
6th Annual CMMI Technology Conference and User Group
Denver, CO
www.ndia.org/

November 13-17
International Testing Certification Week
Orlando, FL
www.testinginstitute.com/stpw/Orlando06

November 15
2006 Software Best Practices Conference
Ft. Lauderdale, FL
www.itmpi.org/events/

2007
2007 Systems and Software Technology Conference
www.sstc-online.org

5. Englehart, K.S., B.A. Hudgins, and P.A. Parker. "Multifunction Control of Powered Prostheses Using the Myoelectric Signal." Intelligent Technologies for Rehabilitation. Eds. Horia-Nicolai L Teodorescu and Lakhmi C Jain. CRC Press, 2001: 1-61.
6. Kuiken T.A., et al. "The Use of Targeted Muscle Reinnervation for Improved Myoelectric Prosthesis Control in a Bilateral Shoulder Disarticulation Amputee." Prosthetics and Orthotics International 28 (2004): 245-53.
7. Strange, K., and J.A. Hoffer. "Restoration of Use of Paralyzed Limb Muscles Using Sensory Nerve Signals for State Control of FES-Assisted Walking." IEEE Transactions on Rehabilitation Engineering 7 (1999): 289-300.
8. Tyler, D.J., and D.M. Durand. "Chronic Response of the Rat Sciatic Nerve to the Flat Interface Nerve Electrode." Annals of Biomedical Engineering 31.6 (2003): 633-42.
9. Rousche, P.J., and R.A. Normann. "Chronic Recording Capability of the Utah Intracortical Electrode Array in Cat Sensory Cortex." Journal of Neuroscience Methods 82.1 (1998): 1-15.
10. Borrisof, J., et al. "Brain Computer Interface Design for Asynchronous Control Applications: Improvements to the LF-ASD Asynchronous Brain Switch." IEEE Transactions on Biomedical Engineering. 51.6 (June 2004): 985-992.
11. Schieber, Marc H., and Marco Santello. "Hand Function: Peripheral and Central Constraints on Performance," Journal of Applied Physiology 96 (2004): 2293-2300.
12. Musallam, S., et al. "Cognitive Control Signals for Neural Prosthetics." Science 305.5681 (2004): 258-262.

Notes

1. Initially, all 16 electrodes were used; to assess performance with eight electrodes, every other electrode was omitted. This procedure was used to assess performance with 4, 2, and 1 electrode.
2. Reproduced with permission of the Rehabilitation Institute of Chicago.
3. Reproduced with permission from the University of Utah and the Journal of Neurophysiology.

About the Authors

Dawn MacIsaac, Ph.D., is currently an assistant professor at UNB where she also directs the software engineering program and is also a researcher with the Institute of Biomedical Engineering. Her areas of research include biosignal processing for rehabilitation and diagnosis as well as software development for biomedical systems. MacIsaac has a doctorate in electrical and computer engineering from UNB.

Department of Electrical and Computer Engineering
University of New Brunswick
Fredericton NB
Canada E3B 5A3
Phone: (506) 451-6968
E-mail: dmac@unb.ca

Kevin B. Englehart, Ph.D., is currently an associate professor of electrical and computer engineering at UNB and is also the associate director of the Institute of Biomedical Engineering. He leads a team that has developed a multifunction myoelectric control system and a real-time embedded prototype of this system. Englehart has served as a consultant for many industrial and government partnerships, including the Department of National Defense in Canada, MacDonald Dettwiler Inc., and Westland Helicopters PLC. From 2004-2006, he served as scientific lead for Diaphonics, Inc. in the development of a speech biometrics product. Currently, he leads a team at UNB participating in two major efforts led by DARPA in the development of a next-generation artificial limb. Englehart had a doctorate from UNB.

Institute of Biomedical Engineering
University of New Brunswick
Fredericton NB
Canada E3B 5A3
E-mail: kengleha@unb.ca

A Brighter Future From Gallium Nitride Nanowires[1]

Dr. Kris A. Bertness, Dr. Norman A. Sanford, and Dr. Albert V. Davydov
National Institute of Standards and Technology

How might gallium nitride semiconductor nanowires change the future of computing? In the spirit of this special issue on how science fiction might become working technology, we offer some speculations and explain the science behind them. This article focuses on what makes nitride semiconductor nanowires unique as technological materials and how those differences could be exploited as the field matures. One major application area is the miniaturization of ultraviolet light sources and detectors for medical devices, forensic and biological analysis tools, ultraviolet-based secure communications, space sensors, mineral identification, and miniature displays. Additional applications include high-power transistors for radar arrays, quantum mechanical devices, and nanoscale mechanical actuators.

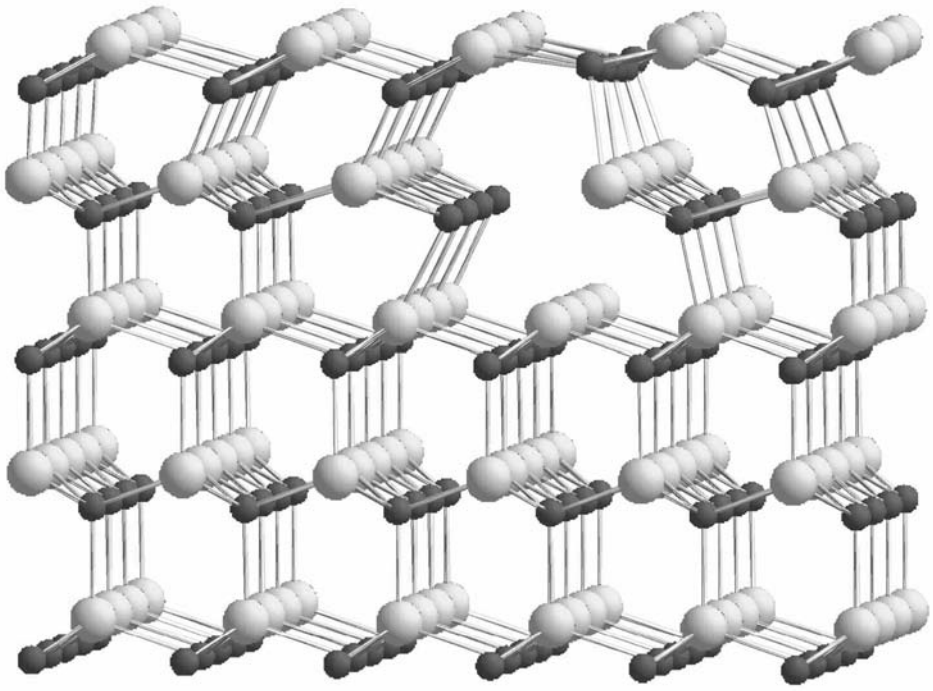
An entire technological revolution began in the 1990s when a series of technological breakthroughs [1, 2] allowed the manufacture of light-emitting diodes (LEDs) and then allowed semiconductor diode lasers that operate at light wavelengths from the ultraviolet to the green. Gallium Nitride (GaN) and Indium Gallium Nitride (InGaN) LEDs are currently a multibillion dollar industry with primary applications in cell phone back lighting, traffic signals, outdoor displays, projection television, and other high-end consumer goods [3]. Nitride LEDs are also poised to displace incandescent and fluorescent lamps for general illumination needs if the manufacturing costs can be reduced. This represents potential worldwide markets of tens of billions of dollars in addition to large reductions in fossil fuel consumption. Blue GaN/InGaN lasers form the technological basis for the higher density DVD storage and playback systems just now being released in select markets and thus impact today's computer technology. Ultraviolet LEDs can also be used for water sterilization and are presently marketed as high-end backpacking equipment. Much of the initial development was funded by the Defense Advanced Research Projects Agency, the Office of Naval Research, and the Department of Energy, with private industry such as Nichia, Nippon Telegraph and Telephone Corp., Samsung, Sumitomo, Cree, Philips/LumiLEDs, Osram, General Electric, Uniroyal, and Agilent all pursuing commercial LED development and manufacture[2].

Limits to Existing Technology

As impressive as these accomplishments and potential markets are, this is only the beginning. Further development is presently limited by cost, yield, and performance issues that are related to the crystal growth process itself. The primary nitride semiconductors are the stable crystalline compounds Aluminum Nitride (AlN), GaN, and Indium Nitride (InN) [4]. In general, it is possible to mix the Al, Ga, and In ratios to make ternary and quaternary alloys such as $Al_{0.5}Ga_{0.5}N$ and $In_{0.2}Ga_{0.4}Al_{0.4}N$. It is, therefore, possible in principle to make semiconductor lasers that emit light from the deep ultraviolet with a photon energy of six electron-volts (6 eV), to the infrared, with a photon energy of 0.8 eV. In practice, only a much narrower range of operation from the near ultraviolet (3.5 eV) to the green (2.4 eV) has been demonstrated. This leaves out a host of applications, including medical and sensor applications where deeper ultraviolet light is essential.

Varying the chemical alloy composition in nitrides induces large changes in the typical atomic spacing in the crystals. Semiconductor devices are generally made by some form of epitaxial growth, which means layer-by-layer growth. In epitaxial growth, deposition starts with a substrate of the same crystal structure desired in the layers. Additional material is deposited under conditions that allow it to copy the substrate crystal structure. If the spacing between atoms in both the substrate and the layer material are equal, this process can go on indefinitely and the growth is said to be *lattice-matched*. If it is not, the layer initially conforms to the spacing of the substrate, but as the layer grows thicker, the strain of maintaining unnatural atomic spacing

Figure 1: *Schematic Diagram of Atoms in a Crystal Structure With a Dislocation Forming to Allow Greater Atomic Spacing on the Top Layers Than on the Bottom Layers. The Larger Light Spheres Represent Ga Atoms, and the Smaller Dark Spheres Represent N Atoms*



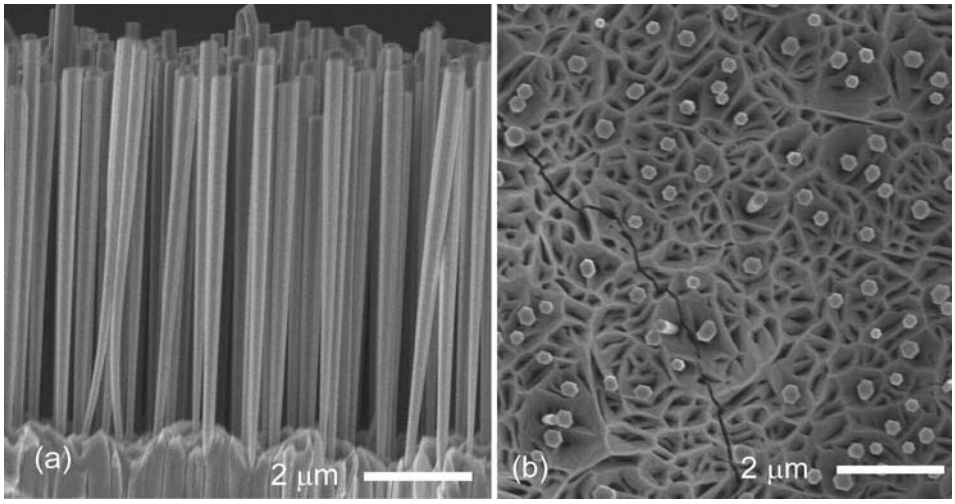


Figure 2: *Field Emission Electron Microscopy Pictures of GaN Nanowires (a) in Cross-Section, Showing the High Aspect Ratio of the Wires, and (b) Top View, Showing the Hexagonal Shape of the Wires*

increases. Finally the layers can no longer tolerate the strain and small atomic-scale defects, called dislocations, form. A schematic diagram of a dislocation is given in Figure 1 (see page 9). Note that the average spacing of the atom rows on the top surface is larger than the spacing at the bottom surface.

These dislocations degrade the optical emission efficiency in LEDs and lead to failure of laser diodes. Dislocations are believed to be the primary factor preventing successful development of ultraviolet lasers, and therefore no diode laser with light emission energy greater than about 3.5 eV has yet been demonstrated. Layer strain can also cause the crystal to separate into regions of different alloy composition, making it difficult or impossible to reach certain alloy combinations. This is a severe design limitation in InGaN devices, where alloys in the middle of the range cannot be grown with conventional technology. Furthermore, there are no lattice-matched substrates upon which to initiate growth in the first place. Unlike silicon wafers, which can be manufactured to high quality with diameters of 300 mm, existing GaN substrates are less than 20 mm across; they are rare, expensive, and high in dislocation density. Commercial GaN devices are typically grown on sapphire or silicon carbide substrates, neither of which have the same atomic spacings as GaN. Even given a successful substrate solution, the variation in atomic spacing within the AlN-GaN-InN alloy system assures that the more interesting combinations of materials will still have to manage strain. This persistent issue for yield and design flexibility is called the *lattice-mismatch* problem.

Nanowires Avoid the Lattice-Mismatch Problem

Enter the nitride nanowire. Nanowires have small cross-sections, allowing them to accommodate much higher levels of strain without formation of dislocations. The wires, furthermore, can rapidly exclude any dislocations that do form to the nearby sidewalls, where they terminate and cease to propagate through the crystal [4]. This defect exclusion mechanism allows the growth of dislocation-free GaN nanowires on the cheapest and highest quality semiconductor substrate around: silicon. Figure 2 shows some typical GaN nanowires grown by the National Institute of Standards and Technology (NIST) by molecular beam epitaxy [5]. The wires range from 50 nanometers (nm) to 400 nm in diameter, depending on growth conditions and can be grown to at least 12 micrometers (μm) in length. The regular hexagonal cross-section arises from the crystal structure of GaN. Although tiny compared with an entire substrate and epitaxial layer used for a conventional semiconductor device, the nanowire volume is still roughly within an order of magnitude of the optically and electrically active region of many such devices. The nanowires can be readily removed from the substrate into solvent solutions and then dispersed onto alternative substrates and processed with photolithography. We have also demonstrated with optical [6] and structural characterization [7, 8] that these nanowires are free of defects and comparable to or better than the best available bulk materials. This high quality persists even when InGaN and AlGaN layers are incorporated within the wires. Other workers in the field have given excellent detailed demonstrations that nitride nanowires can produce higher optical output than conventional materials [9], tolerate large changes in AlGaN alloy composition without defect introduction [10], and reach a broader range of InGaN alloy compositions [11] than is accessible with conventional epitaxial growth technology.

"The nanowire LED and laser will bring new wavelengths and higher energy efficiency to LEDs and diode lasers."

Miniaturization of Ultraviolet Optical Technology: Tricorder, Anyone?

The nanowire LED and laser will bring new wavelengths and higher energy efficiency to LEDs and diode lasers. Nanowires can also be made into wavelength-selective detectors, allowing the miniaturization of spectroscopic analysis. This feature is particularly important in reducing payload size and weight in (real, not fictional) spaceborne sensors and instrumentation. In the future, this technology could conceivably take conventional equipment used for DNA or biological sample laboratory analysis and bring it out into the field. Blue and ultraviolet light are commonly used for fluorescence identification of organic molecules. Current technology relies on power-hungry, high-voltage gas vapor lamps or lasers to produce blue and ultraviolet light and bulky optics to analyze spectra. A handheld, battery-operated (or solar-powered) instrument with functionality such as the *Star Trek* tricorder, then mineral, atmospheric gas, and biological specimen identification could become a reality. Imagine a soldier in remote mountains performing full forensic analysis in real time and uplinking the data via satellite feeds.

Miniature ultraviolet lasers also have a number of medical applications for surgery and diagnostic testing, bringing

the hospitals of the future closer to the fictional sick bays on starships. These types of devices will naturally change the way computers interface with sensors, demanding a more portable and specialized interface and database access.

More down-to-earth applications of visible and ultraviolet LEDs and lasers include the following:

- Data storage with higher spatial resolution.
- Compact low-power systems for water purification.
- Higher speed, chip-to-chip, and board-to-board communications within computers.
- Non-line-of-sight secure optical communication via scattered ultraviolet light.

The last application makes use of the fact that ultraviolet light scatters quite efficiently from particulates in the air. This fact is being exploited for the development of secure UV-based, covert urban combat communication systems over short distances.

Power Transistors for Radar Arrays

AlGaN is presently being explored as a new material system for high-power transistors in radar arrays signal processing because it can operate at higher temperatures and withstand sudden power spikes. Unfortunately, the dislocations that degrade laser performance also degrade transistor performance as well. Specifically, the noise in the amplifiers is increased and the gain factors vary with time and usage. Because nanowires can be made dislocation-free, they should be free of these complications.

Quantum Computing and Communication Devices

Another area where nitride nanowires could have far-reaching applications is in the area of quantum computing. Quantum computers are expected to excel at operations that require massive parallel processing of information, including the important cryptography application of factoring products of large prime numbers. Most of the work in quantum computing with semiconductor devices has been done in the technologically more mature semiconductor system of Indium Gallium Arsenide (InGaAs)/Gallium Arsenide (GaAs). Single-photon sources allow secure communications using quantum cryptography, and commercial systems are available on the market today based on these infrared light sources. The operating wavelengths are well matched for long-distance fiber-optic communications, but at the same time these materials are limited to small band gap energy variations. The devices must be cooled to well below room temperature to operate reliably because these energy barriers are smaller than the typical thermal energies of electrons at room temperature. Alloys in the nitride family have much larger band gap differences. Work on GaN quantum dots and quantum wells is just beginning, and there is promise for both single-electron transistors operating at room temperature and single-photon sources and detectors operating at visible or even ultraviolet wavelengths.

Another advantage of semiconductor nanowires for quantum computing is the inherent isolation and interaction control available in the nanowire morphology. Controlled isolation is necessary for the preparation of quantum mechanical initial states and their interaction without loss of phase coherence. Successful single-electron transistors have been demonstrated in nanowire forms of materials that are otherwise too difficult to manipulate in this way [12, 13]. Actual quantum computation operations and devices are still on a rather distant horizon, but bringing the concept demonstrations up to room temperature will speed progress and lower ultimate operating costs.

"Creating control software for these complex machines will be a challenge because of both the large number of control points and the difficulties in communicating instructions."

Nanoscale Actuators

Unlike other common compound semiconductors such as GaAs and Indium Phosphide, the most stable form of the nitride semiconductors is the wurtzite or hexagonal crystal structure. This crystal structure has a significant asymmetry along one axis, and this asymmetry gives the crystals a large piezoelectric coefficient. Piezoelectric materials expand and contract when voltages are applied along certain crystallographic directions.

Nitride nanowires can thus be used as tiny actuators to manipulate particles on a nanometer scale. Fine tuning of positioning would be useful for cellular and molecular probes and possibly for controlling quantum mechanical interactions (tunneling, for example) between nanowires or quantum dots. Realization of these concepts will require additional technological development in order to make electrical contact to the device in a way that is reliable and easily manufactured. The existing state of the art in electron beam lithography is sufficient for making demonstration devices with multiple contacts to a wire lying on a surface; in fact, most nitride nanowires are long enough to process with conventional photolithography. Progress has also been made in placement and manipulation of nanowires with electric fields and optical tweezers. Functional three-dimensional actuators will require truly flexible electrical contacts to free-standing wire sections, which have not yet been demonstrated but may conceivably be realized by other metallic nanowires bonded to the semiconductor nanowires. Breakthroughs in combining nanowires (and other nanostructures) with biologically based assembly methods or polymer technologies might lead to the ability to create very complex machines – the nanobots of the future – with nanowires providing the muscle. Creating control software for these complex machines will be a challenge because of both the large number of control points and the difficulties in communicating instructions.

Summary

We have outlined some of the major areas of technology in which nitride nanowires could have dramatic impact. The miniaturization and efficiency improvement of blue and ultraviolet light sources will lead to greater portability and broader application. The small size of the nanowires contributes to the possibilities of both quantum device designs and very small mechanical devices. The largest potentials for applications are in computer technolo-

gy, communications, lighting, and health care. This potential has been recognized by *R&D Magazine* editors in selecting our nanowire work as a winner in the 2006 MICRO/NANO 25 competition. All that remains is to *make it so.*◆

Acknowledgements

It is our pleasure to acknowledge the larger team at the NIST who have contributed to this work: J.B. Schlager, L.H. Robins, A. Roshko, T.E. Harvey, I. Levin, M.D. Vaudin, J.M. Barker, P.T. Blanchard, A. Motayed, D. Rourke and M. Greene. We also gratefully acknowledge NIST senior managers who have funded the bulk of our work in this area. NIST is a part of the Technology Administration of the U.S. Department of Commerce.

References

1. Amano, H., et al. "P-Type Conduction in Mg-Doped Gan Treated with Low-Energy Electron Beam Irradiation (LEEBI)." Japanese Journal of Applied Physics Part 2-Letters 28 (1989): L2112.
2. Nakamura, Shuji, et al. "Thermal Annealing Effects on P-Type Mg-Doped Gan Films." Japanese Journal of Applied Physics Part 2-Letters and Express Letters 31 (1992): L139.
3. "Gallium Nitride 2005 – Technology Status, Applications, and Market Forecasts." Report Sc-26. Mountain View, CA: Strategies Unlimited, 2005.
4. Trampert, A., et al. "TEM Study of (Ga,Al)N Nanocolumns and Embedded Gan Nanodiscs." Microscopy of Semiconducting Materials 2003. Institute of Physics Conference Series. Bristol: IOP Publishing Ltd. 180 (2003): 167-170.
5. Bertness, K.A., et al. "Spontaneously Grown GaN and AlGaN Nanowires." Journal of Crystal Growth 287.2 (2006): 522-27.
6. Schlager, J.B., et al. "Polarization-Resolved Photoluminescence Study of Individual GaN Nanowires Grown by Catalyst-Free MBE." Applied Physics Letters 88 (2006): 213106.
7. Bertness, K.A., et al. "Catalyst-Free Growth of GaN Nanowires." Journal of Electronic Materials 35.4 (2006): 576-80.
8. M. Kuball, et al, ed. "High Degree of Crystalline Perfection in Spontaneously Grown Gan Nanowires. GaN, AlN, InN and Related Materials in Materials Science." Mater. Res. Soc. Symp. Proc. Boston, MA 892 (2005).
9. Kim, Hwa-Mok, et al. "High-Brightness Light Emitting Diodes Using Dislocation-Free Indium Gallium Nitride/Gallium Nitride Multiquantum-Well Nanorod Arrays." Nano Letters 4.6 (2004): 1059-62.
10. Risti, J., et al. "Carrier-Confinement Effects in Nanocolumnar Gan/Al(X) Ga(1-X)N Quantum Disks Grown by Molecular Beam Epitaxy." Physical Review B 72 (2005): 085330.
11. Kikuchi, A., et al. "Ingan/Gan Multiple Quantum Disk Nanocolumn Light-Emitting Diodes Grown on (111) Si Substrate." Japanese Journal of Applied Physics Part 2-Letters and Express Letters 43.12A (2004): L1524-L26.
12. Thelander, C., et al. "Single-Electron Transistors in Heterostructure Nanowires." Applied Physics Letters 83.10 (2003): 2052-53.
13. Zhong, Zhaohui, et al. "Coherent Single Charge Transport in Molecular-Scale Silicon Nanowires." Nano Letters 5.6 (2005): 1143-46.

Notes

1. Contribution of an agency of the U. S. Government; not subject to copyright.
2. This is not a comprehensive list; inclusion on this list does not constitute an endorsement by NIST of the products of these companies.

About the Authors

Kris A. Bertness, Ph.D., is a project leader in the optoelectronics division of NIST in Boulder, CO. Her technical focus has been on III-V semiconductor crystal growth, characterization, and device design, including research on nanowires and quantum dots. Prior to joining NIST in 1995, Bertness developed record-efficiency solar cells with a team at the National Renewable Energy Laboratory. She received her doctorate in physics from Stanford University.

NIST
325 Broadway
STOP 815.04
Boulder, CO 80305-3328
Phone: (303) 497-5069
Fax: (303) 497-3387
E-mail: bertness@boulder.nist.gov

Norman A. Sanford, Ph.D., is project leader of the optical materials metrology project at NIST in Boulder, CO. Since joining NIST in 1988, he has been active in a number of research areas including rare-earth-doped waveguide lasers, nonlinear optical characterization of materials, X-ray studies of wide-bandgap III-nitrides, and, most recently, characterization of wide-bandgap III-nitride nanowires. Sanford is a Fellow of the Optical Society of America and received his doctorate in physics from Rensselaer Polytechnic Institute.

NIST
325 Broadway
STOP 815.04
Boulder, CO 80305-3328
Phone: (303) 497-5239
Fax: (303) 497-3387
E-mail: sanford@boulder.nist.gov

Albert Davydov, Ph.D., is head of the semiconductor task group for the International Centre for Diffraction Data at NIST. He has experience in processing and characterization of electronic materials including nanowire, thin film, and bulk crystal growth; structural characterization of nanowires, films, and bulk crystals; metallization of GaN; and experimental and computational study of phase diagrams. Davydov has a doctorate in chemistry from Moscow State University, Russia.

NIST
Metallurgy Division
100 Bureau DR
STOP 8555
Gaithersburg, MD 20899-8555
Phone: (301) 975-4916
Fax: (301) 975-4553
E-mail: davydov@nist.gov

Leadership, The Final Frontier: Lessons From the Captains of Star Trek

Paul Kimmerly
Marine Corps Technology Services Organization

David R. Webb
309 Software Maintenance Group, Hill Air Force Base

Leadership, the final frontier. These are the examples of the Star Trek *captains whose mission was to seek out new methods, to teach management lessons, and to boldly lead where few managers have gone before. Invisible forces, alien invaders, and magnetic storms are all common challenges to the captains of the various ships in the* Star Trek *universe. Managers face their equivalents every day and, while it might be desirable to have Scotty beam you out of danger, it is never that simple. However, managers can learn lessons from the* Star Trek *captains and how they dealt with different situations. The authors look at the advantages and disadvantages in the approaches used by each* Star Trek *captain and how managers can use the lessons taught by the captains.*

As the other articles in this issue of CrossTalk illustrate, the various *Star Trek* series affected more than their legions of devoted fans. The shows inspired technological developments and taught lessons in diversity and diplomacy. To do that, the shows need(ed) strong central characters that audiences can believe. Foremost among the characters were the five starship captains. Each of them had unique challenges, personalities, and approaches to managing their ships. The individual captains brought different styles to their commands, and each of their styles provided examples and lessons for managers in today's world. Let us look at each of the captains in detail and in chronological order as they appeared on our television screens.

James T. Kirk

The first captain to appear on *Star Trek* was an energetic, hands-on leader. He led every crew excursion to new planets and took an active role in all interactions with new civilizations. Captain Kirk also relied heavily on his crew, especially his science officer, chief engineer, and doctor. He pushed them all to succeed but depended on their counsel to help him make decisions. His crew knew who was in charge, but responded to his call for their input and did their best to answer his needs. From Captain Kirk, managers can learn the power of involving and empowering their staff.

Captain Kirk had one talent unmatched by any of the other captains: No one handled being struck by an invisible force like James T. Kirk. Whether it was an energy blow, a psychic blow or some other kind of unseen force, Captain Kirk might double over in pain, but he would push through it to complete his mission. Managers find themselves assaulted by unexpected unseen forces. These forces often cause pain and impact on schedules, staffing, and quality. Like Captain Kirk, managers must find a way through unanticipated problems to reach their goals.

Staffing issues can become major problems for managers. Managers need to ensure that the right people fill the right roles in an organization. Sometimes this includes subordinate managers. In the software world, it is not uncommon to promote an outstanding technician into a management position because they have reached the top of their technical pay scale. While a person may have outstanding technical skills, they may not make a good manager. Captain Kirk had to deal with outbursts from Dr. McCoy that all came down to something like, *Darn it, Jim! I'm a doctor, not a (insert occupation).* Organizations may find themselves hearing the same thing from newly promoted managers. Just as Captain Kirk had to lead Dr. McCoy through those difficult situations, organizations need to mentor new managers through the learning curve of their positions.

The downside to Kirk's method is that his total *hands-on* approach created a management bottleneck – everything had to funnel through Kirk. As McCoy told him in the first movie, *You're pushing, Jim. Your people know their jobs.* The Kirk approach, then, would not work in a diverse environment where workers need to be more autonomous. The Kirk method is more appropriate in a tight, geographically identical team with a culture of strong leadership.

Jean-Luc Picard

Captain Picard commanded a new version of the same starship as Captain Kirk and led with a different style. Captain Picard was a more stoic commander. While he showed humor and compassion at times, he clearly held the position of authority on his ship. His approach made use of his resources in a different way than Captain Kirk. Captain Picard would send an away team to any encounters on new planets. His crew entered the dangerous situations and explorations. They would relay information to the ship where Picard could lead them based on what they provided. Picard showed managers how to gather and use data better than any other *Star Trek* captain. He would collect the data from his away team and then issue an order to *make it so.* That is not to say that Picard was uninvolved. He allowed his people to explore and deal with situations, but he always stayed informed and would act when the time was right. He was less likely to jump into a situation the way that Captain Kirk would but used his staff and information to their best potential.

In opposition to the weakness of Kirk's approach, Picard's *hands-off* approach also had a drawback: While he allowed his staff to stretch and grow and handle all the issues they could on their own, he often kept vital information to himself. From time to time, this created a sense of confusion in the crew as to the captain's intents. Leadership using the Picard style, therefore, is best suited to a large, process-centric, either geographically identical or diverse team, and requires strong communication skills from leadership.

Benjamin Sisco

The commander of the Deep Space Nine base found himself isolated from the mainstream of the Federation and was forced to deal with warring factions. Placed between the Cardassians and Bajorans, Sisco had to be well versed in diplomacy. Taking over the base from the Cardassians, Sisco dealt with the transition from the old rule to the new Bajoran independence. Managers can find themselves in this same situation as projects change over time. It is not unusual for existing software projects to convert to a new technology. This often means starting a project within a project. The old guard and the new project compete for the same

resources and the same attention from the project manager. That manager could learn a lot from the way Sisco balanced the needs of the new Bajoran majority with the needs of the withdrawing Cardassians. Both the new project and the old project need the manager's attention. The old project needs to be assured that its work still has value to the organization while the new project needs the manager's assistance in getting established. The project planning and management knowledge that made the old project successful should be applied to the new project. Doing this ensures that the new project has some structure to its efforts. It also helps to open a communication line between the two projects to ensure the new group learns from the existing group and carries successful approaches forward into the new effort.

Managers who find themselves isolated like Commander Sisco can use that to their advantage by trying new things that may not be as easy to try from the middle of a large corporate structure like the Federation. Managers must keep the organization's goals, objectives, and policies in mind when trying new things, but they also need to experiment and try new methods to fit changing situations. What worked for them before may not work with the next challenge. They must be open to new ideas.

Thrust into the situation at Deep Space Nine, Commander Sisco had a Bajoran as one of his key staff members. Her insight into the problems of her people and their needs helped him in his efforts to manage the situations he faced. Similarly, managers will usually find that the best ideas will come from their staff. Managers should look for opportunities to gather ideas from the staff. The people who perform the day-to-day functions of a project know what works best and what needs improvement. Managers should take every opportunity to get their input.

Most of us who have taken over leadership of a software team can probably identify with Sisco. His greatest weakness was also the source of his greatest strength: inexperience. He was forced to bring order to individuals from diverse groups that brought different skills and biases to the team. As a new leader, he entered the situation with no preconceived notions that could hinder his efforts. Sisco's lessons learned, then, would best be applied to a radically diverse group working in an uncertain environment. A good example would be an integrated project team made up of individuals from different areas within an organization who are brought together to start a new project. The project manager of such a group should always be willing to try new ideas and think outside the station.

Catherine Janeway

The captain of the Voyager faced a unique situation in the *Star Trek* world. She found her ship and crew mysteriously transported across the galaxy to uncharted space. Her mission was to find a way home. Managers of new projects within an organization can feel this same way. This is especially true if the project is something new that the organization has never tried before. In the software world, this often happens with new development projects using new technology. Managers are forced to set their own direction. These projects usually depend on experts in the new technology. Their opinion often drives the course of the project. Managers need to focus this knowledge and expert opinion to meet the project's needs. Captain Janeway relied on the expertise of her crew to deal with resource shortages, equipment needs, and unexpected challenges. She pushed the creativity of her staff to deal with problems. Managers of new projects need to encourage and foster creativity. This will help them find their way to a successful outcome, as the creativity of Captain Janeway's crew helped them find their way home.

"The people who perform the day-to-day functions of a project know what works best and needs improvement. Managers should take every opportunity to get their input."

While Janeway's style may work for other kinds of teams, it is best when directed specifically to small, maneuverable teams. This style, while effective at uniting a small group of individuals to achieve a common goal, may not apply to larger groups.

Jonathan Archer

Chronologically, Captain Archer was the first *Star Trek* captain. He set out on the first Enterprise years before Captain Kirk. Archer faced a number of challenges similar to those of Captain Janeway. While his mission was more clearly defined, he was the first human captain to set out in a starship. He made a number of the rules that the later captains followed. Managers of projects dealing with new technology find themselves in this situation. New, cutting-edge technologies become stable, old technologies. The lessons learned in the first projects with new technologies were vitally important to future successes. All of the captains kept a log that contained their lessons learned in both success and failure. None of these logs was more important that Archer's. He set the standards for the captains that followed. New project managers do the same. Keeping track of their project's challenges and successes, they set the standard for future project managers.

Captain Archer had help in facing his mission of exploring intergalactic space for the first time. The Vulcans were long-time explorers that offered their assistance to the fledgling human explorers. Organizations have managers with project experience. The organization should use these managers to mentor the new projects. If an organization has an established process improvement program, it is likely that there will be lessons learned data from past projects. A process improvement group can also serve the Vulcan role in assisting the new project as it enters the unknown. New project managers should not be hesitant to ask for help. That is what the process improvement group and the lessons learned from past projects are intended to provide.

With even less experience behind him than Benjamin Sisco, Captain Archer was a true pioneer. His lessons learned can be directly applied to new projects taking on brand new work or using new technologies and processes. Team leaders experimenting with Capability Maturity Model® Integration, Team Software Process℠, Agile or other methodologies could rely upon Archer's approach as they blaze a new trail. As with Captain Janeway's method, however, this approach works best in a small, focused team and may break down in a

® Capability Maturity Model and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

℠ Team Software Process and Personal Software Process are service marks of Carnegie Mellon University.

larger, more diverse group.

All Captains

As a group, the captains dealt with unexpected encounters, crew problems, and planetary governments. In a manager's world, these equate to changes, staffing, and customers. All of these issues require decision-making skills and information to help make decisions. The captains can teach lessons in all these areas.

Change is a regular part of a manager's world today. Sometimes, even with the best information, it is difficult for a manager to know how to deal with unexpected and changing situations. When the *Star Trek* captains encountered something unexpected, they put up their ships' shields. Managers often put up their shields when change comes their way, but they should remain open to any situation and not merely dig in defensively. The *Star Trek* captains always reacted cautiously, but they also opened communication channels immediately. Rather than react to changing situations, the captains used open communication to gather information about a situation and determine how to proceed. They realized that not every unexpected encounter or changing situation presented danger; sometimes these situations provided opportunities. Managers can learn that change can enable improvement and innovation.

When it came to dealing with different groups within their crews or from planetary civilizations, the *Star Trek* captains needed to practice diplomacy. Even the more action oriented captains like Kirk and Archer had to learn to negotiate and bring groups together. In organizations, managers deal with customers, both outside the organization and inside other departments within the organization. The *Star Trek* captains provided good examples of when to take a stand and when to work on a mutually satisfactory solution. To guide their actions, all of the captains following Captain Archer had orders to stay within the *Prime Directive*. This Star Fleet edict requires the captains to avoid interfering in the social and technological evolution of any planet and civilization that they meet. For organizations, this shows the importance of establishing a set of goals, objectives, and policies. Guiding principles will help the managers in an organization identify what needs to be achieved in any situation. As a result of following the Prime Directive, Star Fleet showed a common face to the planets and civilizations it encountered. Organizations need to communicate their guiding principles and show consistency in order to avoid differences between their stated goals and the actions of their managers and staff.

Regardless of the situation, the *Star Trek* captains looked to their ships' technology and their crews to gather information before acting. Each ship had sensors to provide the captains with data about whatever ship or planet they encountered. The captains relied on their crews to interpret that data to provide the information needed to make decisions. While they often acted quickly, they did not act rashly. This is an important lesson for managers to follow. Managers should identify their information needs and determine what data is available for them to address those needs. All organizations have available data. The key is for managers to determine what is the most consistent and usable data. Managers should also identify the staff members who understand the data and can provide the best analysis of that data. Once managers receive the information they need, they should use it to make a decision that fits within the organization's guidelines to achieve the organization's goals and objectives.

Conclusion

Each of the *Star Trek* captains faced their own challenges with their own management styles. They all succeeded in inspiring their crews and getting the most from them in sometimes difficult situations. Project managers can learn from the captains and use some of their lessons to be successful. Whether it was dealing with a blow from an unseen force or finding a way home through uncharted space, the captains used their resources and abilities to solve problems and face challenges. The captains provided lessons to today's project managers on how to deal with difficult and changing situations. All of the captains used their resources and available information to make informed decisions to guide their ships. Managers can use similar techniques to lead their projects.◆

About the Authors

Paul Kimmerly has 19 years experience in software development for the different incarnations of the United States Marine Corps Technology Services Organization in Kansas City, Mo. A member of the Software Engineering Process Group (SEPG) since 1993, he has served as the group's chair for the past nine years. Kimmerly is an authorized Standard CMMI Assessment Method for Process Improvement Lead Appraiser. He presented at the 1997 and 2000 Software Engineering Symposiums and the 2004 National SEPG conference, and has contributed several articles on process improvement to CROSSTALK.

United States Marine Corps
TSO-KC/TKGB
1500 E 95th ST
Kansas City, MO 64197
Phone: (816) 926-5364
DSN: 465-5364
Fax: (816) 926-6969
DSN: 465-6969
E-mail: paul.j.kimmerly@dfas.mil

David R. Webb is a senior technical program manager for the 309th Software Maintenance Group at Hill Air Force Base, Utah, a CMMI Level 5 software organization. He is a project management and process improvement specialist with 19 years of technical, program management, and process improvement experience with Air Force software. Webb is a Software Engineering Institute-authorized instructor of the Personal Software Process℠, a Team Software Process launch coach, and has worked as an Air Force flight director, SEPG member, systems software engineer, lead software engineer, and test engineer. He is a frequent contributor to CROSSTALK and has a bachelor's degree in electrical and computer engineering from Brigham Young University.

309 SMXG/520 SMXS
7278 4th ST
Hill AFB, UT 84056
Phone: (801) 940-7005
Fax: (801) 775-3023
DSN: 775-3023
E-mail: david.webb@hill.af.mil

Ten Lessons Learned: Data Warehouse Development Project, California Department of Fish and Game

Crilly Butler, Jr.
California Department of Fish and Game

Despite an abundance of data in numerous internal and external systems, the California Department of Fish and Game struggled over the years to assemble this information into coherent and meaningful representations of its core business. A highly successful data warehouse project has largely resolved this problem, yet the development process had to face and overcome numerous obstacles over the span of several years before all the important lessons were learned. The project was hindered by all the usual culprits: heterogeneous data systems, lack of consistent data standards, limited funds, lack of internal expertise, competing political priorities, and rigid organizational boundaries. Unexpected hurdles included the sudden and serious illness of a key contractor, internal reorganization of business units, and a statewide economic downturn. Each hurdle was met and surmounted in both creative and practical ways, yet widely available within most organizations.

"Water, water everywhere, nor any drop to drink…"

Famous lines by Samuel Taylor Coleridge from *The Rime of the Ancient Mariner*. Could not the same thing be said about most organizations where, despite an abundance of data, staff often go thirsty for information? The California Department of Fish and Game (DFG) has struggled for years to generate comprehensive reports which accurately portray important aspects of its core business. Historically, such reports have had to be cobbled together manually from several data stores – a process that can occupy staff for days or weeks at a time. The recent development and implementation of a large data warehouse with a number of targeted business areas has successfully addressed this problem, yet the sailing was neither smooth nor uneventful. The project sat moored in the harbor for years, got off to several false starts, then sailed around in circles before finally getting its bearings and reaching port. During the project course, DFG was faced with numerous obstacles, both internal and external, yet managed to surmount each with a combination of creative and practical solutions. Perhaps these 10 lessons learned can help others chart a less-perilous course towards the ultimate goal of putting information in the hands of those who need it.

The DFG manages California's diverse fish, wildlife, and plant resources, and the habitats upon which they depend, for their ecological values and for their use and enjoyment by the public. In support of this mission, the department's information technology hardware and network infrastructure have been regularly upgraded to provide most employees with high-speed access to their local data assets and to the outside world via the Internet. Several large, production database applications automate procurement and payments, allocate and adjust the budget, track the sale of hunting and fishing licenses, manage contracts, store information about properties and assets, oversee commercial fishing activities, monitor habitat restoration, and more. Like many other state agencies, DFG also relies on external data processing systems for everything from recording accounting transactions to managing its human resources. These external systems, supplied by control agencies and hosted at large data centers, provide relatively inflexible access to data – generally limited to lookup screens on terminal emulators, voluminous fanfold reports, or downloads of huge fixed-format data files. Direct communication between these external and internal systems is rare, creating the ubiquitous *islands of information* that hinder knowledge management in most organizations.

The need for a centralized, consolidated, departmental data store, optimized for online analytical processing and easily accessible by all employees, was recognized by business analysts at DFG for years. However, implementing such a solution is complex, both with regard to the technology involved and, more importantly, from a business process perspective since it involves the close cooperation of numerous organizational entities, requires dedicated resources throughout the department, and, most critically, depends on active sponsorship at the highest levels – in other words, by people with a lot of other oars in the water.

Initial attempts at data warehouse development stalled at the conceptual level. Though there was widespread support for the idea even among executive management; as a priority, there was little wind in the project's sails. Meetings were held, preliminary analysis was done, and some logical data models were developed but the effort eventually ran out of steam. Part of the problem was a lack of internal expertise in data warehouse development – after all, the department had never attempted it before. This would not have prevented forward progress in and of itself, but when resources were at a premium, and the choice was between *doing the business* or *improving the business*, day-to-day operations generally took precedence. Nevertheless, occasional discussions continued, and when some unexpected funding became available, the decision was made to outsource a project to someone with data warehouse expertise. A contractor was hired, the existing documentation was dusted off, and the initial project was launched.

1. Start Small to Achieve Early Success

The department's internal Budget Management System (BMS), developed as a client/server application some years previously, was woefully lacking in reporting capability, and each request for a new report had to be passed to the programmers for development and deployment. Even tiny reports that would be used only once had to make their way through this same narrow channel.

Deploying a modest data warehouse, both to streamline report development and, most importantly, to put it into the hands of the business users themselves, would be a significant *proof of concept*. Though its scope was small – only a single branch's data was involved – the benefit was immediate.

2. Optimize the Design

Budget branch staff were recruited to participate in the development of the

warehouse, along with the department's own analysts. After detailed discussions, the contractor produced a denormalized, business-centric *star schema* (see Figure 1). This is a standard data warehouse design with one or more *facts* tables comprising the hub of each star, surrounded by various *dimension* tables that allow the level of granularity of the *facts* to be drilled into or rolled up along relevant vectors. For example, budget *allotment facts* could be viewed along different *time dimension* slices so that allotment adjustments, as they evolved from day to day or week to week, could be visualized. Or, alternatively, budget allotment *facts*, and their specific impacts on various programs or funds, could be either summarized or detailed throughout the entire program or funding hierarchies. Or, lastly, budget allotment *facts*, as they are distributed throughout the department, could be viewed with respect to any level of the organizational hierarchy.

This is quite different from the fully normalized relational architecture adopted by most online transaction processing (OLTP) systems. The *star schema* pays no consideration to data redundancy since quick response time is the highest priority. Because the schema is denormalized, accessing and filtering huge data sets is extremely fast since table joins are kept to a minimum and data access paths are fully pre-planned in accordance with the current business needs. Robust indexing has no impact on performance during *insert*, *delete*, or *update* transactions since only *select* transactions by end-users are permitted. This also eliminates the kinds of insert, delete, and update anomalies inherent within denormalized designs and ordinarily resolved by a normalized relational architecture.

3. Implement in Increments

The information technology branch (ITB) staff granted the contractors read-only access to the production client/server database so that the OLTP data could be imported into the warehouse using an export/import tool bundled with the department's relational database management system (RBDMS). These kinds of tools were generally able to accept as a data source nearly any form of exported file, either in text or binary format, including comma-separated values, fixed-length/fixed offset data, extended binary coded decimal interchange code, etc. They could then reformat the data on the fly and import it into the relational database tables in fairly flexible ways.

Once the data was loaded, a set of preliminary reports was generated to permit the knowledge experts to vet the warehouse data against the source data. Problems and discrepancies were identified and reconciled in an iterative process spanning several weeks. Some were the result of misunderstandings about the data and data relationships, and others were due to mistakes that were made during imports and exports.

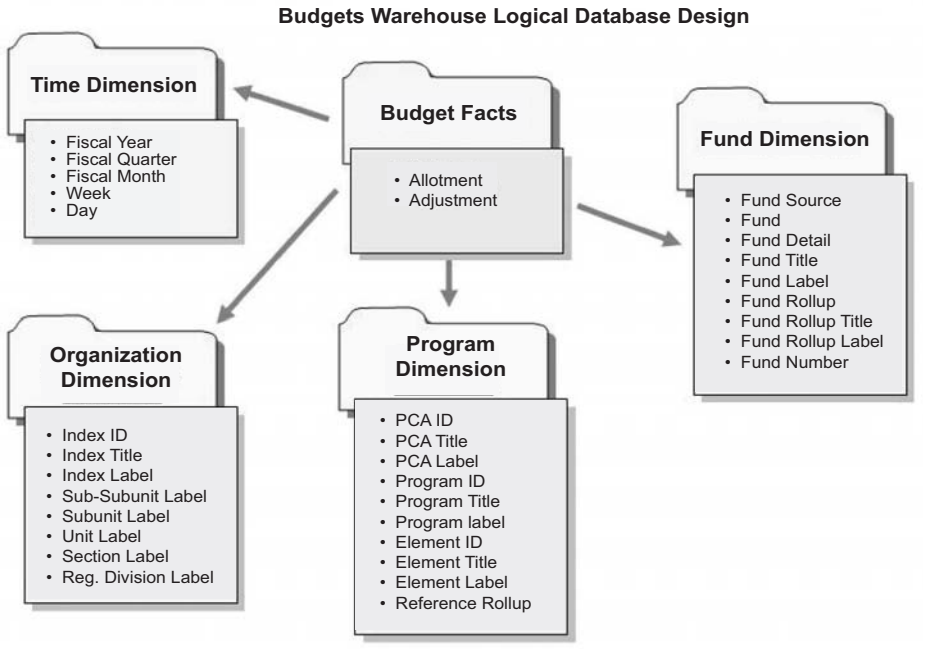


Figure 1: *Budgets Warehouse Logical Database Design*

Finally, when the warehouse data and the production data were seen to consistently match, a nightly, automated replication of the data into the warehouse was scheduled, tested, and confirmed. Over time, the budget branch staff began to use the warehouse with increasing confidence, and for the first time since the inception of the BMS, its data was now available to everyone in the branch in an organized, intuitive and easy-to-access manner.

4. Seize Opportunities

Due to this preliminary success, a second, more ambitious project was planned. The new warehouse would allow for the accurate and timely tracking of expenditures and encumbrances against allotments, permitting regional and divisional staff to know precisely where they stood, at any time, in managing their year-to-date budget. Unfortunately, the currents driving this new effort soon shifted and the project was essentially scuttled. The primary contractor had become gravely ill, the initial funding was nearly exhausted, and internal reorganization led to the remaining resources being redirected elsewhere. More importantly, the enhanced scope of this new warehouse project was obviously going to require collaboration among several branches within the department, such that managers within each branch would have to commit significant staff time to the development effort and would have to make themselves available for meetings and interviews. In light of everything else that was happening in the department, this kind of commitment was simply not forthcoming.

Fortunately, this same departmental reorganization produced a new business unit with staff who had the rare combination of both strong technical skills and business expertise – perfect for taking charge of a data warehouse development project. Additionally, the state's economy was suffering through a downturn, and the department's executive management was being required to submit numerous fiscal reports to both the legislature and the governor to present plans for budget and staffing cutbacks. These reports were difficult and onerous to create, given the scattered and heterogeneous nature of the department's data. During good economic times, state departments can think in terms of dollars, but during the kind of lean times DFG was facing it had to scramble for every penny. The need for a fiscal data warehouse, in which budget allotments could be compared directly with detailed and accurate year-to-date expenditures and encumbrances, sliced and diced through any cross-section of the department's internal structure, became an imperative, and the payoff also directly benefited all the business units required to participate. The

Assistant Deputy Director for Administration said *make it happen*. Thus, the project acquired an admiral, and the ship finally picked up some steam.

The recently formed Business Analysis Unit (BAU) was thus asked to create a fiscal data warehouse that combined allotment data from the internal BMS with financial transactions (expenditures and encumbrances) from CalStars, California's statewide accounting system, hosted by the Department of Finance (DOF). No additional resources could be made available, and there was no budget for training, outsourcing, or hiring of consultants. Still, a great deal had been learned during the initial project that could be effectively leveraged for this one.

5. Involve End-Users Throughout

The importance of good working relationships between all the stakeholders was clear from the earlier development effort, but we learned that it was not enough to simply hold interviews, gather requirements, and then build a system that meets them. Small but noticeable problems with the initial project were recognized to be the result of insufficient involvement by business unit staff at every stage. Input and feedback from those who would ultimately use the warehouse were seen as necessary throughout the entire development process, leading not only to the satisfaction of basic requirements, but just as importantly to the satisfaction of the users themselves. Consequently, management from each unit was asked to recruit their most appropriate knowledge experts to actually join the development team.

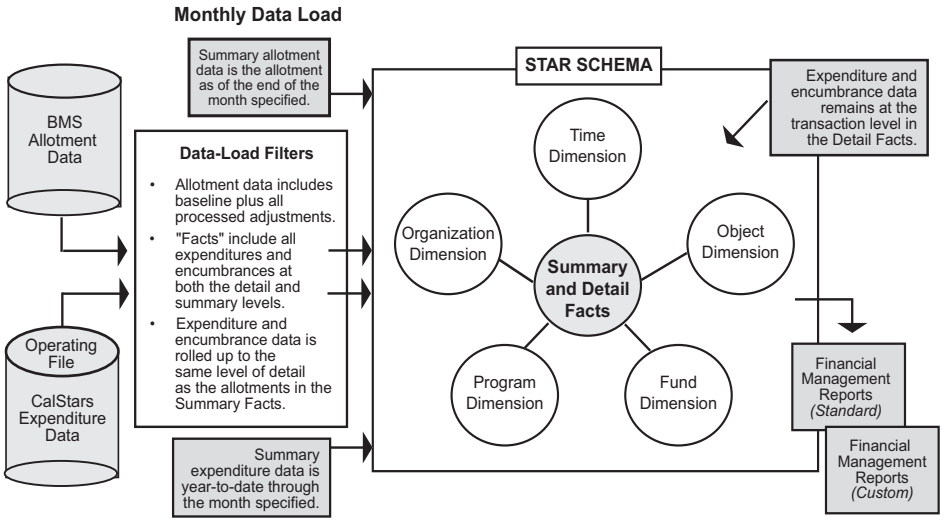
Discussions began in earnest between key staff from the department's budget branch, accounting branch, and BAU staff. ITB was also engaged, even though the actual development work would be done by the BAU since a fair amount of database administration would certainly be required in addition to planning for and accommodating the increased demands on servers, storage, and the network as a whole. Executive management was also consulted regularly to make sure that their reporting requirements could be fully met by the proposed design.

New *facts* and *dimension* tables were sketched out and held up before the business users for their input and suggestions. Again, BAU staff worked closely with those who would be using the warehouse until consensus was reached with regard to the essential facts to be focused on, and the critical *dimensions* along which the data needed to be manipulated.

DOF, host of the CalStars accounting system, made available two large binary data files at the close of each fiscal month – one containing year-to-date totals for transactions posted against all funds as of the close of that month and one containing all the individual transactions posted that month. Correspondingly, the requirement became obvious for two basic sets of *facts* – allotments, expenditures, encumbrances, and account balances from both a *year-to-date* perspective (cumulative amounts), and a *current month* perspective (individual transactions). The first set, *summary facts*, would enable managers to see exactly what they had been allotted to spend, and exactly what has already been spent or encumbered, year-to-date, from any account. The second set, *detail facts*, would allow managers to examine each expenditure individually, looking for erroneous charges or overcharges or to explain exactly why the *year-to-date* totals looked the way they did. Several discreet dimensions would enable these dollar values to be analyzed along the axes of time, organizational entity, program, fund, or object (expenditure category).

Budget allotment data would then be exported from the department's internal RDBMS and merged with the accounting data from CalStars. Since the accounting data was stored at a lower level of detail than the budget allotments, accounting transaction dollars had to be rolled up to this higher level in the *summary facts* table to make comparisons valid (see Figure 2).

Figure 2: *Allotment-Expenditure Warehouse High-Level Design*



6. Make It Business Friendly

Once the logical warehouse design was agreed upon and in close consultation with the business users, the physical database was developed in such a way that all the *facts* and *dimension* tables and data elements were assigned names based on their common usage within the department. We discovered one characteristic of a well-designed data warehouse is that everything in it should be immediately recognizable and intuitively meaningful to those who will use it. In addition, by honoring user input in this way, a certain *pride of ownership* among the business unit staff develops that is generally lacking in many software development efforts, not to mention the feeling of respect that this inherently communicates to staff.

Another important feature was to build in flexibility so different users could visualize the data in ways that are meaningful to them. For example, those who were familiar with coded values, such as A1772, should continue to be able to do so, generating concise reports that gave them exactly what they needed. At the same time, managers – more familiar with complete titles – should be able to see *San Joaquin River Salmon Telemetry* instead. Similarly, those dealing with fiscal year data and those basing their reports on calendar year data are easily accommodated.

Lastly, *facts* and *dimension* tables should be designed so that they can be prejoined in only a single way, eliminating the need for users to have to understand the underlying data structures or be required to make technical decisions when choosing which data to display in their reports.

7. Add Value Wherever Possible

Because data from two different systems was to be merged together in the warehouse, it soon became obvious that we

had to reconcile the different representations of the same data. This led to a concurrent, long overdue, and ultimately successful effort to create some department-wide data standardization. This was an unintended benefit, since the need for such reconciliation only became apparent during these first attempts to combine data from the two systems. Data standardization had been talked about within the department for some time and seen as a *desirable goal*, but no one had ever been given the responsibility for *data administration* to provide this oversight or spearhead this effort. Since there was no way to successfully merge the two data sets without reconciling the differences, the BAU was assigned *de facto* responsibility to become the data administrator for the department. Since disparities were significant but not huge, the effort had only minimal impact on the overall development timeline, yet it provided important value to the department, both in the current system and for future development efforts.

Conversations to understand and reconcile data variations involved the owners or stewards of each set of data plus the BAU analysts. Simply drawing attention to these disparities was an eye-opener to many who had no idea that other units were representing identical information quite differently. Where identical codes with disparate meanings were being used by different organizational entities, a consensus was reached as to which version would be kept and which modified, cascading those modifications out to all other affected systems. Where the same codes were used to represent the same data, but where the titles varied somewhat, the more complete or descriptive title was chosen as the standard. Abbreviations, viewed as a holdover from legacy systems and their limitations, were generally eliminated as unnecessary in the belief that full names produce fewer misunderstandings and less confusion.

8. Provide a Simple Implementation

Each new fiscal month's data would be loaded into the warehouse by the BAU just after the close of the fiscal month. A link to the Web-based user interface was then deployed to the appropriate section of the department's intranet home page by the ITB's Web Services Administrator – a simple HTML change pointing to the correct server and directory. The data warehouse interface itself was provided by a Web service that comes bundled with the department's RDBMS. This tool, being Web-based, can be made easily accessible to any employee able to connect to the department's intranet and with an appropriate database account, without the need for time-consuming and difficult-to-maintain software installations. First-time users were required to download a browser plug-in that enables the browser to communicate with the database, but this installation is triggered automatically during the first login session and never needs to be repeated.

Within six months, the warehouse had been designed, developed, and deployed. Not only was the data now available on the desktop to everyone who needed it, but because the lengthy and tedious distribution of paper reports throughout the state was no longer necessary, the new data were available weeks earlier than ever before. Budgeting and procurement decisions could now be based on and supported by detailed and accurate information. The warehouse development project had evolved into an unqualified success.

9. Maintain Momentum

There is a saying that quickly became applicable to this project: *supply breeds demand*. No sooner was the fiscal data warehouse available, than employees throughout the department began clamoring for more. The business users quickly understood the power of the data warehouse concept and began wanting the same kinds of access to other areas of the department's data.

If a fiscal data warehouse could be developed so quickly and easily, why not develop one for each of the following:

- Covering all departmental contracts.
- Developing labor cost accounting that brings together personnel data from the state Controller's Office with time sheet hours and charges from the CalStars accounting system.
- Reporting revenue.
- Reconciling staff positions from Human Resources with position funding from the BMS.
- Tracking and reporting on procurements and invoice payments.
- Monitoring vehicle usage.

In other words, staff had glimpsed an alternative to isolated and difficult-to-access islands of information and wanted consolidated, business-centric data available to them immediately, regardless of how many distinct and heterogeneous physical systems on which the production data actually resides.

10. Support Your Users

As demand escalated, the introductory information covered in the initial round of presentations was no longer sufficient to meet the complex needs of business users. They wanted greater access, training, documentation, and improved performance. So a warehouse primer was created, with step-by-step instructions for accessing the warehouse, running canned reports, modifying existing reports, creating new reports from scratch, sharing reports with other users, and exporting data to spreadsheets for additional manipulation. More comprehensive two-day trainings were scheduled throughout the state to provide business-unit-specific instruction and business-unit-specific reports. New dedicated servers, optimized for an online analytical processing environment, were procured to decrease wait-time and improve the user experience. The business-user response could not have been more positive or appreciative.

Ten Lessons Learned

Despite encountering many difficulties, enduring numerous delays, and proceeding with only limited resources and a small budget, the DFG has evolved a successful data warehouse development methodology centered around 10 lessons learned. In summary, they are the following:

1. Start small to achieve early success.
2. Optimize the design.
3. Implement in increments.
4. Seize opportunities.
5. Involve end-users throughout.
6. Make it *business friendly*.
7. Add value wherever possible.
8. Provide a simple implementation.
9. Maintain momentum.
10. Support your users.

Once the decision was made to move forward, the likelihood of success was maximized by constraining the scope of the initial project to a small and workable scale. Producing the first, modest product and demonstrating its immediate and important benefits, gave impetus to future efforts to expand our data warehouse development.

Starting with a design optimized for data warehousing maximized the quality and value of the initial product while laying the foundation upon which future efforts could be built, layer by layer, in sensible and incremental efforts.

While obstacles were encountered, and resources ebbed and flowed, opportunities did present themselves, and it was obvious that each, no matter how small or fleeting, had to be seized and exploited

if development were to produce the maximal benefit.

The department's most valuable asset, however, consists of its hard-working and dedicated employees and their knowledge and understanding of both the business and the data. This became the project's greatest resource. Project developers learned, from the inception, to involve knowledge workers and stakeholders at every stage and to work closely with them to ensure a usable and effective product. At the same time, active executive sponsorship at the highest levels was found to make the ultimate difference between moving forward or floundering. Though this support initially seemed to hinge on the scale of the crisis, resulting from not having access to the data that this project would provide, the long-term benefits soon became clear and executive managers became strong supporters of ongoing efforts.

It was also important to produce a resource that was *business-centric* rather than *information-technology* centric. This maximized usability, acceptance, and value to those who would need to depend on it for gathering and making sense of the department's huge stores of data, and it also served to demystify the data and data relationships.

We learned that mining data also uncovered heretofore hidden data anomalies and discrepancies within the organization. Rather than reacting to this with surprise or distress, it was seen as a valuable opportunity to launch efforts to better manage and administer the department's data assets, improving its consistency and standardization across diverse business areas.

The *simple is better* axiom was also confirmed when it comes to implementing these projects. Our thin-client, Web-based interface, immediately accessible by anyone on the network, was low in cost, simple to maintain, and quick to deploy. Perhaps not as robust as the client/server version of the same tool, this was more than compensated for by its many virtues.

Early successes can quickly be forgotten if momentum is not maintained. Fortunately, the groundswell within the department for more was immediate and energetic. By responding quickly with helpful documentation, on-site training, ongoing phone and e-mail support, and the continual creation of new and varied custom reports, forward progress rarely faltered.

Conclusion

DFG has now developed seven discreet business areas in its comprehensive data warehouse, each addressing one or more of the needs described earlier. Some are in full production, others are in pilot rollout, and two are still in development. A new and ambitious project to assimilate all the human resource branch data into a new business area is in the planning stages. At this point, the creation of new business areas in the data warehouse has become almost routine. Scores of canned reports have been written, each carefully named so that its content is clear and unambiguous. Business unit staff are requesting help building custom reports almost daily and many, within a short time, have become adept at producing their own reports without assistance. The time and costs associated with printing, copying, and distributing hard copies throughout the department have been dramatically reduced as users now simply log on to the warehouse, highlight a report, and run it.

The need for making data available to the people who require it is clear and ever-present. In many cases, vital facts and figures that used to take days or weeks to collect and organize are only a mouse-click away. Now that the ship has arrived with its cargo of information, the department's employees are *thirsty* no longer – at least for today.◆

About the Author

Crilly Butler, Jr. is a senior information systems analyst with the California Department of Fish and Game. A licensed psychotherapist, he switched careers when he realized that a computer would never call him at 2:00 a.m. threatening to jump. Butler specializes in database and data warehouse design and development, along with information systems and business analysis, leveraging his past training in interpersonal communication to *build bridges*. A previous software project he led was accepted into the Smithsonian's permanent research collection in 2000.

California Department of Fish and Game
1416 9th ST
RM 1248B
Sacramento, CA 95814
Phone: (916) 651-7844
Fax: (916) 651-8763
E-mail: cbutler@dfg.ca.gov

A System View of Merging Software and Hardware

Mike McNair
Science Applications International Corporation

No single example is universally applicable in describing the merging of hardware and software into an integrated system. Likewise, no single recipe can be given for making the effort a successful one. In spite of this, there are generalizations that can be identified. The observations presented in this article and the subsequent recommendations are intended to make the effort of merging hardware and software a successful one.

For most of us who have endured a hardware-software integration (HSI) effort, we can all attest to the fact that not only was there technical challenge but programmatic challenge as well. From an engineer's perspective, maybe management just really did not understand what support was needed, that schedules needed to include contingency time, and that there just might have been the added cost of one more analyzer.

There is a flip side, however. From the manager's view, could the tools have not been anticipated, how much schedule allowance is really needed, and maybe more to the point, how could the risk be properly assessed and subsequently mitigated?

It is very easy and somewhat simplistic to think of merging software and hardware as a purely technical issue; instead, it is a mix of technical, programmatic, and personnel interrelationships. In order to understand the interrelationships between these areas, it is important to recognize what merging hardware and software *is* and *what it is not*.

Hardware/software merge activities are not limited to just the following:

- Activities during integration. They really exist throughout the entire development cycle.
- Embedded applications or real-time systems. This applies to being able to move data from a bar code scanner to a database as well as controlling a radar system.
- Device drivers and interrupt handlers. These facilitate the interface but do not form the complete description of the interface.
- Reliance on interface definition. But instead, a successful allocation of functions to satisfy the requirements for a system.

To make it more clear, the following are examples of a hardware/software merge activity:

- Models and algorithms (software) have a physical impact (hardware). Otherwise, software is just an algorithm and the hardware is just a machine
- Engineering disciplines share common ground. This means common milestones, common understandings, and common requirements.
- Demos, prototypes, and products are made possible until the point at which the system is typically depicted as a collection of models, simulations, and emulations.

As a general observation, I have seen many software engineers grow frustrated with trying to trace a software bug only to find it was really a hardware flaw. Likewise, I have seen hardware engineers grow frustrated with a software engineer who does not understand the physical limitations of the device. Sometimes you just cannot make the hardware do everything software can or vice versa.

> *"It is very easy and somewhat simplistic to think of merging software and hardware as a purely technical issue; instead, it is a mix of technical, programmatic, and personnel interrelationships."*

Characterizing a Hardware/Software Project

Regardless of how a project is characterized and modeled, it is usually possible to identify some simple and high-level phases that constitute its life cycle (iterative, spiral, waterfall, etc.). For the purpose of this article, we will assume the presence of these phases since these can usually be found in some form in any life-cycle model: requirements analysis, development, integration, and test/verification.

Each of these phases has a technical and management component that plays a part in the overall effort of merging hardware and software. These technical and managerial activities can be described by a set of activities and characteristics. The following is a look at those activities as they pertain to merging hardware and software.

Requirements Analysis

During the *requirements analysis* phase, it is generally expected that concepts will be pinned down, requirements will be identified, and top-level allocations of functions will be made to subsystems and possibly between hardware and software components. Management is usually busy setting customer expectations and managing the natural influx of ad-hoc requirements. It is important to realize that the customer as well as the product team will be sources of capabilities that are not originally within the product's scope.

Development

Development is almost always planned and implemented with a discipline or specialty focus (product team, hardware/software, logistics/training, etc). Work naturally gets segregated to the groups that make up the development organization. As a result, *stovepiping* is not only common but is considered *normal*. It is typically very difficult to integrate these groups during development whether from a cost and schedule perspective or by an engineering task. Management is usually busy facilitating information flow and keeping the independent disciplines coordinated. It is during this time of a project that emphasis is placed on schedules and milestones; there is little else (aside from status meetings) that a manager has available to keep these groups coordinated.

Integration

As happens with any project, the day

comes when the products of the development effort have to fit together. Whether the pieces have been integrated in an iterative or spiral type approach or done all at once as a part of a waterfall life cycle, there is a real incentive to make the system work. Technically, if all goes well, initial indications are that the software seems to make the hardware *do something*; yet, if problems are encountered, they can be difficult to isolate and fix. From a management perspective, there can be very real cost and schedule impacts – either making up lost schedule when integration goes well or losing schedule when solutions seem difficult and elusive.

Test/Verification

Finally, the test and verification team has an opportunity to determine what the integrated whole actually is able to do. The engineering team is reminded of the requirements, and an outside team is able to assess how effective the effort to date has been at providing the product as it is specified. As the results are unveiled, the functions become capabilities and the problems become limitations. Management is again working with the customer to reset expectations and to define the measures of success. If success can be properly defined, the product is released and put into use.

The Big Picture

Along the way and with each program phase, software and hardware have been merged. If we review the development phases with this in mind, we begin to understand what role merging hardware and software has in the overall process.

In the *requirements analysis* phase, allocations are identified between hardware and software. The processors for the software to run on are assessed and selected. The interfaces to the different devices are specified, which in turn results in design decisions for the software. As algorithms are developed and models are created, hardware selections are made to achieve necessary performance. During the *development* phase, these allocations are used to allow detailed work to progress. Interfaces are identified, defined, and refined. The effort is managed in terms of the work packages that result from these allocations. During the *integration* phase, these same interfaces and allocations are used to identify hierarchical levels of testing with an eye toward increasing levels of integration with each successive level. Finally the product undergoes test and verification. Internal interfaces in essence *go away* and the external interfaces characterize the system. Internal interfaces are seams of the product that either strengthen the set of capabilities or cause stress points where the system can be broken.

Merging Hardware and Software Successfully

From the experiences of all of this, there are some valuable principles that can be identified. With each new project, the hope is for the disciplines to work together better and to act in a more integrated fashion. There is a push for resources and risks to be identified and assessed through lessons learned. It is hoped that the development environment and test tools can be standardized to increase familiarity and make the whole engineering effort run more smoothly and enhance the creativity of the engineering staff. While there are many ways to document these principles (certainly the following list is not comprehensive), these principles and observations can make the merging of hardware and software more productive.

"While it is good to take ownership of the assigned task, it must be remembered that functions as well as defects are a characteristic of the product and not the person."

- **Observation One:** *Interfaces are viewed differently by different groups.* Systems engineering uses interfaces as a means of identifying allocation boundaries and defining allocated component interactions (not behavior). Development engineering uses interfaces as a means of determining *what is mine* and *what is yours* for the purpose of bounding the solution space. The integrators use interfaces as the means of identifying components and how they *bolt* together. Here, interfaces are not just boundaries; interfaces are how expected functionality is scoped. The test and verification team relates to interfaces depending on where they are: Internal interfaces are the seams where the product is most likely to break, whereas external interfaces create customer perceptions and expectations.
- **Observation Two:** *Software and hardware groups use interfaces differently.* These two groups see each other in a very unique way, but it all boils down to one basic premise: Software executes on hardware and hardware is exercised by software. This principle is exemplified where hardware and software merge. The software team sees interfaces in terms of messages, queues, services, etc. Interfaces, including where hardware needs to be controlled, are a part of an abstract model that is depicted in code and data models. The hardware team sees interfaces in terms of wiring, connectors, drivers, etc. Interfaces are a means of achieving connectivity with other components in the system. The usual pitfall is that neither group focuses on system-level functions – both groups tend to lose a system view and the role they play in the overall system capability. It is not sufficient to know the interface; it must be understood that the interface is a piece of the whole system.
- **Observation Three:** *People personalize the components.* When hardware and software do not merge easily, it is easy to fall into the traps of *I said/you said, the interface spec is wrong, we are dealing with bad requirements*, and other finger-pointing type phrases. Engineers discuss components in terms of *I will send you the message* or *with this signal, I will enable an actuator*. Personalities become an inherent part of the system. In short, people take ownership of parts of the system and personally associate with it. While it is good to take ownership of the assigned task, it must be remembered that functions as well as defects are a characteristic of the product and not the person.
- **Observation Four:** *Hardware/software allocation is becoming more a difference in performance than functionality*. As a project moves through requirements analysis and into design, functional requirements are allocated to some combination of hardware and software for implementation. Before the introduction of Application Specific Integrated Circuits, Floating Point Gate Arrays, microcontrollers, and other special purpose processor targets, the allocation approach was simple. Now, instead of just relying on general purpose processors that execute application specific code, algorithms can be

embedded on the processor itself, thus bringing in the added option of application specific hardware. Identifying the proper allocation of a system function can now be through software or application-specific hardware. The initial factor in making this allocation decision usually rests with satisfaction of the system performance requirements.

- **Observation Five:** *Not everyone is adept at merging hardware and software.* Every member of the project team brings experience and training to apply to the problems they are assigned to solve. These differences in experience and training generally result in some people having a knack for requirements analysis over design, or test over integration. It is especially important to recognize this when selecting personnel to conduct the integration effort. When software and hardware meet, the integrator cannot necessarily be someone who just happens to have some spare time. Integration is not usually viewed as the *fun* part of a project. In fact, people are usually recruited from an engineering group or integration is assigned to the test team. Neither of these provides the independence and immersion in the effort that is required.

  If the effort cannot be staffed with someone focused on the effort of making hardware and software meet and work together, the risks increase, and as the risks are realized the schedules stretch, the engineering teams lose their effectiveness and the costs seem to skyrocket. Someone must be responsible for managing an engineer's desire to get results and a manager's need for trade-offs and options. That person has to understand that there is a decision and not a solution.
- **Observation Six:** *A properly specified and defined interface is not the* end all *to successful integration.* Integration cannot rest completely on the definition of the interfaces. The interfaces were identified in the first place through an act of allocating capabilities to systems and components – both hardware and software. If the allocation is wrong or ineffective, no amount of interface definition will help the system. Interfaces, when they are identified, are expressed in terms of software interfaces (possibly documented in Interface Requirement Specifications and Interface Description Documents) and hardware interfaces (similarly documented in Interface Control Drawings). What is missing is the role they play in overall system functionality and the identified system components. In the final product, internal interfaces existed to provide interconnections. If they cannot provide this between properly allocated systems and components, integration (including merging hardware and software) will be very painful to accomplish.
- **Observation Seven:** *Vendor support is beneficial.* There are times when you just need to find an expert who can help you build your system. Usually at a time when schedules, budgets, and patience have all been stretched thin, calling in help is not at the top of the to-do list. The usual fears abound of unplanned cost, uncertain return, lack of domain knowledge, etc., but these are combined with the realization that money is running low, the delivery date is quickly approaching, and there are still significant problems to solve. Admittedly, calling in a consultant or some kind of niche expert brings its own set of risks.

  The better solution is to use these experts along the way, whether from your tool vendor or as a consultant. These subject matter experts (SMEs) tend to have very specific exposure to problems and solutions that the more general development is likely not to have. Allowing an SME to provide a third-party assessment of technical approach can be a great source of implementation alternatives as well as a source of lessons learned. Keep in mind that a vendor has probably seen many more applications of their products than either you or anyone else on your team. Make use of their expertise through training, phone calls, e-mail, Web support, and even on-site support when necessary.

"Every member of the project team brings experience and training to apply to the problems they are assigned to solve."

Closing Thoughts

The observation that stands out the strongest and the lesson that is relearned the most is that hardware/software projects are tough and as such demand careful and realistic planning and execution. While every system and application is different, one thing is clear: Merging hardware and software crosses teams and disciplines. In order to be successful, all members of the project team must acknowledge their part in working as an integrated team that will deliver an integrated product.◆

Background Reading

1. Walker, Royce. Software Project Management: A Unified Framework. Addison-Wesley, 1998.
2. CMMI Product Team. "Capability Maturity Model Integration Version 1.1 Continuous Representation." Pittsburgh, PA: Carnegie Mellon University, Mar. 2002.
3. Project Management Institute. "A Guide to the Project Management Body of Knowledge (PMBOK Guide)." 3rd ed. Newton Square, PA: Project Management Institute, Oct. 2004 <www.pmi.org>.

Additional Reading

1. SoftwareProjects.org <https://www.softwareprojects.org>.
2. Software Program Managers Network <http://www.spmn.com>.

About the Author

Mike McNair is a senior systems engineer for Science Applications International Corporation where he is a part of the chief engineer team for unmanned ground vehicles on contract to the U.S. Army. His background includes more than 20 years of experience as a programmer, technical lead, and software program manager on projects ranging in size, complexity, and targets for a variety of customers. McNair has also served on several process improvement (including Software Engineering Process Group lead) and training initiatives.

8303 N Mopac EXPY
STE B-450
Austin, TX 78759
Phone: (512) 366-7834
Fax: (512) 366-7860
E-mail: mcnairmk@saic.com

A Gentle Introduction to Object-Oriented Software Principles

Maj. Christopher Bohn, Ph.D., and John Reisner
Air Force Institute of Technology

For many of today's software professionals, the concepts behind object-oriented (OO) software principles are as familiar as the tips of their fingers; for others, this is not the case. If you are a structured-programming developer who is beginning to suspect that OO is not simply the flavor-of-the-decade, or if you are the program manager of a software-intensive system who has never written code, you may be wondering: Just what is this OO paradigm? This article will help you find that answer.

The notion that an object is *an instantiated member of a class* is a comfortable and familiar one to Generation X programmers who were educated in the 1990s and to programmers with Ada95, C++, or Java experience. However, to those who have backgrounds replete with Pascal, COBOL, or FORTRAN skills, the OO paradigm can be intimidating, confusing, or even exasperating. If you examine OO code, the constructs seem familiar enough: declarations, loops, functions, and procedures. So, what about OO development is so vastly different, unique enough that it even has been designated as a new paradigm?

In a nutshell, the OO approach:

> ... encapsulates data with corresponding operations and employs polymorphic mechanisms such as class inheritance. With data encapsulation, object classes can be conveniently reused, modified, tested, and extended. [1]

This quote, while a concise summary of the paradigm, is laden with OO-speak, providing little help to the curious neophyte.

This article explains several key aspects of OO software – abstraction, encapsulation, information hiding, decentralized problem solving, inheritance, polymorphism, and reuse – with the aim of presenting a clear overview and summary of the unique strengths of object-oriented software development.

Abstraction

Abstraction is a term used to describe putting certain parts of a system in a black box; that is, to ignore the implementation details of that subsystem. We frequently use abstraction in everyday life: We start our automobiles without thinking about ignition systems or internal combustion, we send documents to printers without fully understanding spoolers and queues in the network.

With abstraction, system developers do not need to think about systems in their full complexity. It would be the rare automobile designer who could simultaneously keep track of details about the exhaust manifold and the seat belt buckle; similarly, software developers need to *abstract-away* details of some parts of the system while focusing on others. Also, with abstraction, code can be reused without full knowledge of the reused component. Instead, we can focus on an abstract mental model of the component's expected behavior, which we will refer to as its *cover story* or *contract*. As long as reused code maintains the cover story, and as long as it does not violate its contract, it does not matter how its task is accomplished. Again, consider the automobile designer: Subsystems that rely on the correct operation of an engine can be designed to work regardless of whether ignition timing is accomplished with a mechanical distributor or by using an electronic timing mechanism. So long as the timing works correctly, the cover story is not broken, and the system will work.

Telling programmers that they *do not need to know* implementation details might make them uneasy. Should they not be concerned about reliability, supportability, and efficiency? Absolutely. Abstraction is not used to integrate shoddy code into a system under the guise of reuse. Rather, we do not *need to know* the details because the contracts already provide sufficient information about the expected object behavior. The end goal is to have a software architecture that facilitates both substitution and reuse. This also helps the system evolve with minimal *collateral damage* from changes (*i.e.* we can change one part of the system without having to change others, an important and recurring theme in OO).

Encapsulation

Reuse is nothing new – code reuse libraries have been around for decades. But in OO, we reuse more than algorithms; we reuse classes.

In OO, classes and objects are closely related; an *object* is a particular member (or *instance*) of a *class*. Classes are a kind of template, providing a means to define both the *state* and *behavior* of an object.

For example, a car's *state* might be described as *engine running, traveling east at 45 mph, with a total of 150 miles traveled*. We might expect the possible *behaviors* of a car to include turning the engine on, turning the engine off, moving at some speed and direction, changing the speed and direction, and retrieving the total distance traveled.

Once the object's state and behavior are described, we can then describe a *class* of cars. This is done by identifying a car's *attributes*, as well as the methods (sometimes called the *operations*) that a car can perform. An example of this is shown in Figure 1.

In an OO software system, an Automobile class is likely to look quite a bit different than the one shown in Figure 1. After all, Defense Advanced Research Projects Agency (DARPA) Grand Challenge [2] aside, computers do not change the direction and speed of a car; drivers do. However, consider a hypothetical information system for a state's bureau of motor vehicles. Such a system might also model a class for automobiles, but it would look more like the one depicted in Figure 2.

An object is an *instance* of a class. When we create a new instance, the attributes of that particular object will take on specific values. For example, if someone bought a brand-new Lexus as a gift for their spouse (like on those holiday commercials), that would necessitate a trip to the license bureau, where a friendly clerk would *instantiate a new object* of the Automobile class type (actually, the clerk would do this unknowingly – thanks to abstraction – merely by clicking on a New Registration icon). For this instance, Make = Lexus, Model = GS300, Year = 2007, Color = Silver, etc. The attributes of a class are analogous to a *record* in languages such as Pascal, while the methods of a class are implemented as procedures and functions. Collectively, the values of the attributes define the object's *state*. Essentially, the attributes and methods jointly implement the class' contract: They specify what information the object contains, as well as its expected behavior.

OO systems place the data (the attributes) and the operations that use that data

(the methods) together. Moreover, the object's state can only be changed by using the operations defined in the class. Organizing software systems in a modular way is not new [3], but this *encapsulation* of attributes and methods – that is, the grouping of data and behavior in the same entity, as opposed to them being dispersed throughout the system – is a key differentiator between the structured and object paradigms.

Information Hiding

As a rule, when we define a class, we specify that the methods are *public* while the attributes are *private*; that is, objects can call the methods of another object, but no other object can have direct access to the values of its attributes. This means that the only way to read or change the state of an object (*i.e.*, to read or change an attribute's value) is through the class' methods.

In our Figure 2 example, we can assume that the Register method will assign initial values to all of the attributes (the first six likely entered by the clerk, while the seventh might be automatically generated with a calendar function, e.g., *one year from today's date*). The renew( ) method will only affect one attribute, plateExpirationDate. If some other scenario requires other read or write access to the variables, then methods to perform these reads or writes would need to be added to the class. For example, we might want to add a changeColor( ) method, so that if an auto owner repainted his car, that change could be reflected in the system without having to re-register the vehicle. However, there would not be a changeYear( ) method, as the model-year of an automobile would not change.

Using the methods to access the attributes means that the information is *hidden* by a *public interface*, (*i.e.*, the data are hidden behind the public methods of the class). If the information were not hidden, then another object could directly access and change an Automobile object's year, either accidentally or by design causing that object to break its cover story (the modified contract states that Automobiles can be registered, renewed, and re-painted, but not artificially aged). Returning for a moment to the real-world automobile (Figure 1), anyone who suspects that a screwdriver has been used to tamper with a car's odometer can sympathize with the perils of direct access to data that should be protected.

In the realm of OO, we can assume that any part of a component we depend upon will always behave as expected, and that no other part of the system will cause our part to violate a contract. Encapsulation and information hiding keep both of these assumptions safe.

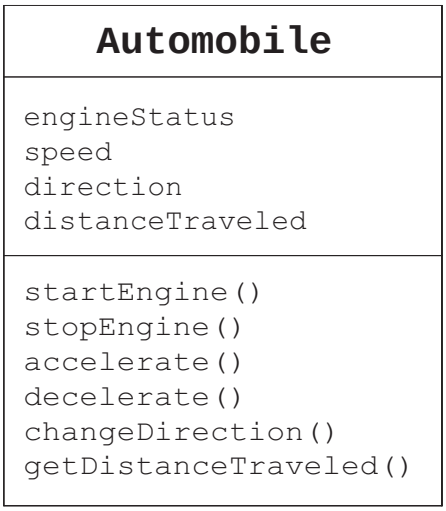
Decentralized Problem Solving

Thus far, our examples have been limited to a single class (i.e., the Automobile class). In a small-scale system, encapsulated objects are a nifty idea, but the full benefits of these assumptions are not realized until a system with many interacting classes is designed.

OO software uses the concept of *message* passing to create an architecture in which objects request services from and provide services for each other. When you want an object to do something, you *invoke one of its methods* by *passing a message*. This is very similar to making a procedure or function call. However, there are two notable differences, one being physical (the called code resides in a different and separately compiled module from the calling code); while the other is conceptual (the method invoked is part of a class' predefined public interface). These differences allow us to design systems that have interacting classes of objects with well-defined responsibilities. In a vehicle registration system, such an architecture may not yield much of benefits, but in other applications (such as a windows-based operating system, or a suite of gaming software), the potential benefits are quite high.

Perhaps more importantly, the object paradigm allows us to create decentralized solutions to computing problems. Decentralization permits us to solve complex problems that we could not solve with a centralized solution. For example, consider the post office. If the clerk behind the counter was responsible for delivering every incoming piece of mail, then the postal service would be very inefficient indeed. Instead, the post office employs classes of people and machines with well-defined responsibilities – clerks, sorters, handlers, deliverers – each doing their part to deliver around 100 billion pieces of mail annually [4].

Figure 1: *A Class of Automobiles, Defined By State (Four Attributes) and Behavior (Six Methods)*



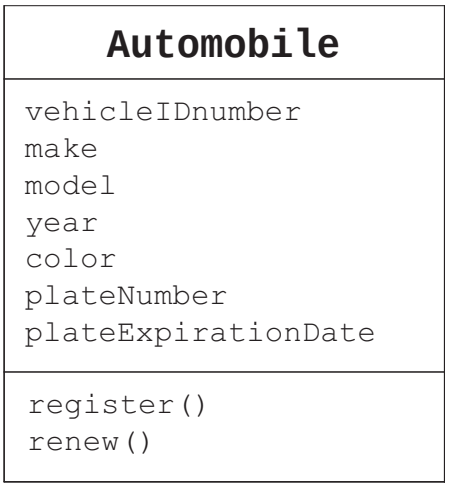
Inheritance

In the course of developing OO software, we often find that we need a class of objects that *specializes* an existing class. For example, a Leased Automobile is a special type of automobile. Besides the make, model, and other attributes that every automobile has, a leased automobile has another attribute: the lessor. Another special type of Automobile is a state-owned automobile. Through a mechanism called *inheritance*, we can formalize this relationship. We call Automobile the *parent class* or *superclass*, while LeasedAutomobile and StateAutomobile are known as *child classes* or *subclasses*. This is sometimes called the *is-a* relationship: a LeasedAutomobile is-a kind of Automobile. Everything that is true about a parent class is also true about its child classes; the special types of automobiles have all the same attributes and operations as a standard automobile. Plus, they have additional attributes and/or operations. For example, a Leased-Automobile also has a Lessor attribute, and a StateAutomobile has a scheduleMaintenance( ) operation. The attributes and operations in Automobile need not be reproduced in its subclasses; they are *inherited*.

Another situation is known as *generalization*. For example, boats also need to be registered. If we discover that there are many similarities between boats and automobiles, we can create a new class, Vehicle, and make Automobile and Boat subclasses of Vehicle. The commonalities among Automobile and Boat are then moved into the Vehicle superclass, possibly renaming attributes that represent the same concept by different names.

Inheritance leads to improved reuse and maintainability. Because subclasses inherit all

Figure 2: *Another Automobile Class, One More Likely to be Found in an Information System*



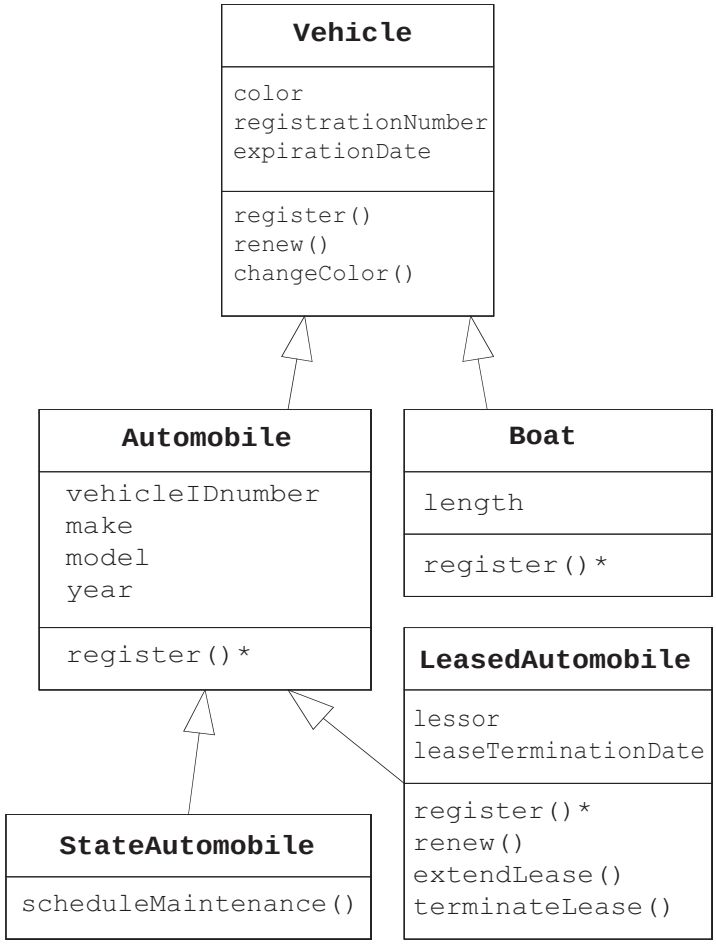


Figure 3: *An Inheritance Hierarchy of the Vehicle Class*

the attributes and operations of their superclass, they automatically get all the debugged functionality that is already in their parent class. When creating a child class, we need only concentrate on the ways in which the new class is different from its parent.

Maintainability improves because when a change is made to a parent class, then the subclasses (and *their* subclasses) automatically get the changes, too. Make the change in one place, and the change is propagated through the entire inheritance hierarchy. Another way maintainability is improved is through isolation of the subclasses. If a change needs to be made for only one specialized class, then those changes can be made without any risk of introducing new errors into other classes. This increased maintainability is even more beneficial as code evolves over time.

Polymorphism

Because everything that is true about a parent class is true about its child classes, instances of the child classes will maintain their parent's cover story as their own. They will have a richer cover story, to be sure, but they should not break the cover story they inherited. To put it another way, their contract is an amendment of their parent class' contract: They may promise a little more, but they will also honor their parent's contract.

This is the beginning, but not the end, of the concept known as *polymorphism*. Recall that one of the purposes of abstraction is to support the substitution of one component with another that honors the first component's contract. Polymorphism is a special form of substitution, in which the component to be used may not be known until the program is running.

There is more to polymorphism than simply substituting one class with another that has a few extra attributes and methods. When we create a subclass, we can *override* some of the methods in the superclass. That is, the method has the same name and parameters as the method in the parent class, but actual code for the method is different, providing the behavior appropriate for the particular subclass. Consider the inheritance tree in Figure 3; we have marked with an asterisk the operations that are overridden in the subclasses. The register( ) operation has been overridden by Auto= mobile, Boat, and LeasedAutomobile so that the clerk would be prompted to enter values for the additional attributes. StateAutomobile, though, reuses the register( ) operation provided by its parent class. Similarly, all classes except LeasedAutomobile reuse the renew( ) operation provided by Vehicle, since they do not need any special behavior. LeasedAutomobile, however, overrides renew( ), since the lessor may need to be consulted.

When the software expecting a Vehicle object is provided an object whose specific class is somewhere in Vehicle's inheritance hierarchy, the operations appropriate to that object will be invoked automatically (the details of how this happens are well beyond the scope of this paper). For example, if an Automobile object is registered, the code found in the register( ) method of the Automobile class is executed. However, when a LeasedAutomobile object is registered, then the register( ) code executed is the overriding method found in Leased Automobile. This selection of the correct method to be executed occurs automatically, without any embedded if-then-else or case logic required; therein lies the power of polymorphism.

Tying It All Together

The concepts presented in this article are often difficult to grasp, especially in a single sitting. Moreover, people who are just beginning to understand these concepts often find that they can follow an OO designer's line of reasoning, but have a more difficult time when it is their turn to try it out. Some authors have said that it takes six to nine months of experience to become proficient in fully exploiting OO techniques [5]. Further, the impact of adopting an OO mindset is not limited only to design; indeed, it has implications throughout the development life cycle, from requirements analysis to design, to implementation and test, and on through maintenance. Volumes could be (and have been) written on each of these topics. To become more familiar with the breadth and depth of OO development, we recommend that you investigate educational opportunities from a reputable continuing education provider, and that you find a software engineer with mature OO skills to serve as a mentor as you grow into increasingly larger, increasingly critical endeavors.◆

Additional Reading

If you are interested in learning how the OO approach influences the software life cycle, you may want to start with these books. Alistair Cockburn's *Writing Effective Use Case*s can help you turn your functional requirements into use cases, which will drive the rest of your project's development; Doug Rosenberg's *Use Case Driven Object Modeling with UML: A Practical Approach* does a fine job of describing the

activities you would use to begin your design from those use cases. Implementation issues can be language-dependent, but Steve McConnell's *Code Complete* and Timothy Budd's *An Introduction to Object-Oriented Programming* address the issues as language-neutral as possible. Paul Jorgensen's *Software Testing: A Craftsman's Approach* is a good book for testing in general, and the last five chapters specifically cover object-oriented testing. Similarly, Thomas Pigoski's *Practical Software Maintenance* covers the breadth of software maintenance, with a chapter dedicated to OO's impact. If you're already familiar with OO concepts, you may want to look at some more advanced books, such as Ken Pugh's *Prefactoring*, Joshua Kerievsky's *Refactoring to Patterns*, and David Astel's *Test-Driven Development: A Practical Guide*.

References

1. Lee, G. Object-Oriented GUI Application Development. Prentice-Hall, 1993.
2. DARPA. Grand Challenge <www.darpa.mil/grandchallenge/>.
3. Parnas, D.L. "On the Criteria To Be Used in Decomposing Systems into Modules" Communications of the ACM 15.12 Dec. 1972: 1053-1058.
4. United States Postal Service. "2005 U.S. Postal Service Annual Report" <www.usps.com>.
5. Eckel, B. Thinking in Java. Prentice-Hall, 2002.

About the Authors

Maj. Christopher Bohn, Ph.D., is a software engineering course director at the Air Force Institute of Technology's (AFIT) School of Systems and Logistics where he teaches a series of distance-learning short courses. Over the past 13 years, Bohn has served in various Air Force operational and research assignments, including time as an engineer for the Air Force Research Laboratory's Collaborative Enterprise Environment. He is an Institute of Electrical and Electronics Engineers-certified software development professional. Bohn has a bachelor's degree in electrical engineering from Purdue University, a master's degree in computer engineering from AFIT, and a doctorate from Ohio State University.

AFIT/LS Research Park Campus
3100 Research BLVD
Kettering, OH 45420-4022
Phone: (937) 255-7777 ext. 3415
DSN: 785-7777 ext. 3415
Fax: (937) 656-4654
DSN: 986-4654
E-mail: christopher.bohn@afit.edu

John Reisner is the Director of Extension Services at the Air Force Institute of Technology's (AFIT) School of Engineering and Management. He retired from the Air Force in 2005 after serving 20 years as a software developer, systems analyst, and instructor of Software Engineering. He is an Institute of Electrical and Electronics Engineers-certified software development professional and has a bachelor degree from the University of Lowell and a master's degree from AFIT.

Air Force Institute of Technology
2950 Hobson WY
Wright-Patterson AFB, OH
45433-7765
Phone: (937) 255-3636 x7422
DSN: 785-3636
E-mail: john.reisner@afit.edu

More Online from CrossTalk

CrossTalk is pleased to bring you this additional article with full text at <www.stsc.hill.af.mil/crosstalk/2006/10/index.html>.

What Science Fiction Authors Got Wrong, and Why We re Better Off For It

Maj Christopher Bohn, Ph.D.
Air Force Institute of Technology

Computers are commonplace today, and they have changed much of the way we go about our lives, saving time and money. Surprisingly, as much as classical science fiction authors were visionaries, they did not foresee the arrival of the digital computer and its influence on modern life.

Robert Heinlein once had a discussion with a professional astronomer in which the conversation turned to Heinlein's attention to detail. Heinlein recounted how, when writing Space Cadet (1948), he and his wife used yards of butcher paper over three days to calculate a particular orbit; the results were described in the story with one line of text, but it was necessary to drive the drama. When the astronomer wondered why Heinlein did not use a computer to make the calculation, he replied, *My dear boy, this was 1947*.

Of course, computers are commonplace today, and they have changed much of the way we go about our lives, saving time and money. Surprisingly, as much as Heinlein and other Golden Age science fiction authors were visionaries – Heinlein, for example, is credited with inventing the waterbed and tele-operated manipulators (waldoes), and with heavily influencing spacesuit design; none of them foresaw the coming of the digital computer and its influence on modern life.

So while computers are everywhere today, Heinlein did not have ready access to computers and neither did his characters. In *Starman Jones* (1953), faster-than-light travel is possible by entering hyperspace singularities on precise vectors. Computing course corrections to get the spaceship on the correct vector required astrogators to make rapid calculations making use of mathematical tables and a computer that amounted to little more than a 4-function calculator. Heinlein, a former naval officer, no doubt was drawing an analogy with mathematical tables classically used to navigate seaships (Interestingly, Charles Babbage was inspired to develop a mechanical computer in the 19th century while reviewing errata for a set of mathematical tables for celestial navigation).

All We Need to Know About Software Project Management, We Can Learn From Watching Star Trek

David R. Webb
309 Software Maintenance Group, Hill Air Force Base

Are you Kirk, Spock, Sulu, Chekov, Uhura, or Scotty? If you are part of a modern-day software engineering team, chances are you fit one of these roles. As we have read about Star Trek*'s influence on technology, sit back for a warp drive look at how a software engineering team's structure mimics that of the* Star Trek *bridge crew.*

A few months ago, I watched a cable TV show which asserted that a *Star Trek* actor had *changed the world*. While the idea that William Shatner had single-handedly brought about the 21st century as we know it was funny, the information presented was pretty convincing that the science fiction show had a major impact on modern technology. Just call your bank's voice-recognition computer using your flip-phone and … well, you get the picture. And it does not end with the original series; those reconfigurable, flat-panel touch screens that Geordi and Data sit in front of in *The Next Generation* can now be found in just about every fast food restaurant!

As I marveled at the effects a television series had had on our modern lives, I suddenly realized that most of the current software generation had grown up with *Star Trek*. I began wondering what effect this had on our approaches to software development and management. What I discovered I have termed *The Gene Roddenberry Effect*: Everything we do in software project management originated with *Star Trek*. I have developed a list of lessons learned from *Star Trek* that I regularly employ.

The Bridge Crew

One surprising example of how *Star Trek* has affected software project management is how the composition of the bridge crew on the starship Enterprise reflects that of current software teams. In the original series, most of the stories centered around the bridge crew: Kirk, Spock, Sulu, Chekov, Uhura, Scotty (you know the names). Kirk was the ultimate leader and decision-maker, Spock was second-in-command and science officer, and all the others had specific job titles as well.

Modern industry has pretty much borrowed this structure for software engineering teams. Every project has a project manager (PM), a technical expert who acts as backup PM, a chief engineer, a communications officer (configuration manager), and so forth. The PM calls the shots, much like the captain (Kirk), and everyone else performs their particular jobs and regularly reports back to him. When the team has issues, it gathers in a conference room and projects everything on a flat panel monitor for all to review. The team makes a group decision, guided and ratified by Kirk. Watch those conference meetings in the original series and see if you get goose pimples at how similar they are to the meetings you go to each week.

Perhaps most surprising is the similarity between the bridge crew concept in conjunction with the Software Engineering Institute's (SEI) Team Software Process℠ (TSP℠). The TSP is SEI's *how-to* guide for implementing high-maturity software engineering teams. In the TSP, each member of the team is given one of eight management roles. As I began looking at them, I was amazed at how closely they matched the bridge crew concept (see Table 1).

What is even more interesting is that many TSP teams actually do double up on roles as shown in Table 1. If they do not have eight people to perform the different roles, people take on more than one.

But wait; it gets even better.

Once we move to *The Next Generation* series, we find a whole new bridge crew with expanded roles. In this case, Picard, the captain, is no longer the PM, but has risen to the rank of *senior management* (as described in SEI's Capability Maturity Model Integration 1.1). We can see from Table 2 that the team concept has matured aboard the newer Enterprise and that our more modern approaches to the software team concept have mimicked this structure.

We could go even further and examine the newer *agile* development methodologies only to discover that as the *Star Trek* series matured, so did their concepts of teams and agility. Captain Janeway's Federation/Maquis team aboard Voyager, for example, is the epitome of an agile team. In any case, whether or not it was done intentionally, it seems that modern software teams are indeed modeled after the *Star Trek* bridge crews. Some of the agile methods even measure a project *velocity*. A coincidence? Well, maybe.

Table 1: *TSP Roles vs Enterprise Bridge Crew*

TSP℠ Roles	Bridge Crew Role	Bridge Crew Member
Team Leader	Captain	Kirk
Customer Interface Manager	Communications Officer	Uhura
Design Manager	Science Officer	Spock
Implementation Manager	Chief Engineer	Scotty
Planning Manager	Navigation Officer	Sulu
Process Manager	Operations Officer	Chekov
Quality Manager	Medical Officer	Bones
Support Manager	Science Officer	Spock
Test Manager	Chief Engineer	Scotty

Lessons Learned

So, with the obvious links between the *Star Trek* universe and the way we manage software projects, what else we can learn from *Star Trek* that can actually help us in our day-to-day management of software projects? The following are the 10 project management tips I use most often:

10. When push comes to shove, it's always a computer geek who comes to the rescue! (Watch just about every *Star Trek: The Next Generation* episode with Wesley Crusher.)
9. Your greatest challenges can be the ones you thought you got rid of 200 years ago. (Remember Kahn [Ricardo Montalban] from the original episode

Space Seed and the movie *Star Trek II*? Remember Y2K?)

8. If everyone on your project is too happy, you probably are not accomplishing anything. (Yes, it's the hippie episode about the spores, called *This Side of Paradise*. For those who are not ardent fans, trust us on this one.)
7. You cannot change the laws of physics, but you *can* bend them. (This is a line made famous by Scotty in the original series episode entitled *The Naked Time*.)
6. There is no such thing as a no-win scenario, especially if you change the conditions of the test. (This is Kirk's philosophy and fits pretty well into his profile above! Watch *Star Trek II – The Wrath of Kahn* for how this worked for him … or did not.)
5. Every successful project manager has both a good side and bad side and knows how to balance them. (Let's just hope you do not beat yourself up the way Kirk did in *The Enemy Within* – literally. In that episode, Kirk had an *evil twin* created by a transporter malfunction.)
4. Don't feed the Tribbles or they will overrun you! (Substitute your own set of problems for Tribbles as seen in *The Trouble with Tribbles*. If you do not know what a Tribble is, why are you reading this article?)
3. If you do not apply a lot of power to break away from your routine, you are doomed to repeat the same mistakes over and over again for eternity. (*The Causality Loop* from *The Next Generation* – great episode! Go rent it!)
2. Some people can be very afraid of change. (Let us just hope it doesn't lead to the shedding of blood, red or pink as it did in the movie *Star Trek VI – The Undiscovered Country*.)
1. Always multiply your estimates by a factor of four so that you will be known as a *miracle worker*. (This comes from the movie *Star Trek III – The Search for Spock*, and we have no comment as to why this is number one or how we implement it; especially if any of our customers are reading this!)

TSP℠ Roles	Bridge Crew Role	Bridge Crew Member
Senior Management	Captain	Picard
Team Leader	First Officer	Riker
Customer Interface Manager	Security Officer	Worf
Design Manager	Science Officer	Data
Implementation Manager	Chief Engineer	Geordi
Planning Manager	Navigation Officer	Wesley
Process Manager	Operations Officer	Data
Quality Manager	Medical Officer	Crusher
Support Manager	Ship's Councilor	Troi
Test Manager	Chief Engineer	Geordi

Table 2: *TSP Roles vs The Next Generation Enterprise Bridge Crew*

Conclusion

While this article is a bit tongue-in-cheek, it's amazing how closely our software engineering practices mirror *Star Trek*. So live long, prosper, and remember to thank (or maybe curse) the *great bird of the galaxy* next time you put together those PowerPoint slides for your management review!◆

About the Author

David R. Webb is a senior technical program manager for the 309th Software Maintenance Group at Hill Air Force Base, Utah, a CMMI Level 5 software organization. He is a project management and process improvement specialist with 19 years of technical, program management, and process improvement experience with Air Force software. Webb is a Software Engineering Institute-authorized instructor of the Personal Software Process℠, a Team Software Process launch coach, and has worked as an Air Force flight director, Software Engineering Process Group member, systems software engineer, lead software engineer, and test engineer. He is a frequent contributor to CROSSTALK and has a bachelor's degree in electrical and computer engineering from Brigham Young University.

309 SMXG/520 SMXS
7278 4th ST
Hill AFB, UT 84056
Phone: (801) 940-7005
Fax: (801) 775-3023
DSN: 775-3023
E-mail: david.webb@hill.af.mil

Web Sites

The National Nanotechnology Initiative

www.nano.gov

The National Nanotechnology Initiative (NNI) is a federal research and development (R&D) program established to coordinate the multiagency efforts in nanoscale science, engineering, and technology. The goals of the NNI are to: maintain a world-class R&D program aimed at realizing the full potential of nanotechnology; facilitate transfer of new technologies into products for economic growth, jobs, and other public benefit; develop educational resources, a skilled workforce, and the supporting infrastructure and tools to advance nanotechnology; and support responsible development of nanotechnology. Twenty-three federal agencies participate in the Initiative, 11 of which have a R&D budget for nanotechnology. Other federal organizations contribute with studies, applications of the results from those agencies performing R&D, and other collaborations. The NNI is managed within the framework of the National Science and Technology Council (NSTC), the cabinet-level council by which the President coordinates science, space, and technology policies across the federal government. The Nanoscale Science Engineering and Technology (NSET) subcommittee of the NSTC coordinates planning, budgeting, program implementation and review to ensure a balanced and comprehensive initiative. The NSET subcommittee is composed of representatives from agencies participating in the NNI.

The Science of Star Trek

http://ssdoo.gsfc.nasa.gov/education/just_for_fun/startrek.html

Is *Star Trek* really a science show, or just a lot of *gee, whiz* nonsensical sci-fi? Could people really *do* the fantastic things they do on the original *Star Trek* and *Next Generation* programs, or is it all just hi-tech fantasy for people who cannot face reality? Will the real world come to resemble the world of unlimited power for people to travel about the galaxy in luxurious, gigantic ships and meet exotic alien beings as equals? What are the features of *Star Trek* that a person interested in science can enjoy without guilt, and what features rightly tick off those persnickety critics? Well, many of the star systems mentioned on the show, such as Wolf 359, really do exist. Usually, though, the writers just make them up! There have also been some beautiful special effects pictures of binary stars and solar flares which were astronomically accurate and instructive. The best accuracy and worst stumbles can be found among the features of the show that have become constant through all of the episodes. The site contains a list of the standard *Star Trek* features, roughly in order of increasing scientific credibility.

Star Wars Technology, Coming Soon to a Planet Near You

http://science.nasa.gov/newhome/headlines/sc19may99_1.htm

Science fiction is the infinite realm of what might be, sometimes just a few minutes into the future. The *Star Wars* movies flash-dozens of futuristic concepts past the viewer's eyes, but how likely are these concepts? Some might be closer than you think. Check the possibilities on this site and click to the stories about the research that NASA is conducting today to make *Star Wars* technology happen tomorrow.

American Association for Artificial Intelligence

www.aaai.org

Founded in 1979, the American Association for Artificial Intelligence (AAAI) is a nonprofit scientific society devoted to advancing the scientific understanding of the mechanisms underlying thought and intelligent behavior and their embodiment in machines. AAAI also aims to increase public understanding of artificial intelligence, improve the teaching and training of AI practitioners, and provide guidance for research planners and funders concerning the importance and potential of current AI developments and future directions. The major sections of this site (and some popular pages) can be accessed from the links on this site. To help you choose, they have included short explanations within the links themselves.

How William Shatner Changed the World

www.discoverychannel.ca/on_tv/how_shatner/shatner_home/

Could a TV show propel us into a real-life final frontier? Could a fictional captain change the way we interact with our world? While you could argue that William Shatner changed the world, he is downright sure of it, and he has taken the liberty of making a two-hour documentary to prove it. From communicator-style flip-phones, to medical imaging, to space-craft propulsion, *How William Shatner Changed the World* reviews how the fiction of the U.S.S. Enterprise and crew inspired a world of science in reality. Click on some of the *influences* headings on the site to read the technology's timeline and see if you're convinced that the technical revolution is because of one man.

The Star Wars Worlds: More Science Than Fiction?

http://news.nationalgeographic.com/news/2005/06/0603_050603_starwars.html

Everyone knows the *Star Wars* galaxy is located "far, far away." But how realistic are the alien worlds described in the science fiction saga? To find out, National Geographic News checked in with two experts on everything extraterrestrial: Bruce Betts, a planetary scientist at the Planetary Society in Pasadena, CA, and Seth Shostak, a senior astronomer at the Search for Extraterrestrial Intelligence Institute in Mountain View, CA.

Technovelgy

www.technovelgy.com

The Technovelgy Web site gives visitors the opportunity to explore the wide variety of inventions and ideas of science fiction writers. More than 1,050 of these inventions are shown on this site, including the progress made on each invention. Visitors can view the online timeline of inventions derived from science fiction, beginning with the Geometric Modeling – 18th century NURBS (from *Gulliver's Travels* by Jonathan Swift) and ending with External Eyelenses (from *Altered Carbon* by Richard Morgan). Doing a search for Star Trek and Star Wars technology, a visitor can find more than 40 inventions from Star Trek or Star Wars that have either become a reality or are currently being developed. Invention topics include, but are not limited to armor, artificial intelligence, biology, clothing, travel, vehicle, virtual person, warfare, weapon, and work.

Science Fiction, Science Fact

There exists a magic weapon that terrifies enemies and is capable of unparalleled destruction. There is no known defense, nor effective countermeasure. This secret weapon, when deployed, causes mass destruction and terror – terror so great that the opposing military commander's best advice is to advise his troops to *get down on our elbows and knees, and beseech Our Lord to save us from this danger*[1]. Science fiction or science fact?

Science fact. This magic weapon was Greek Fire. It was a liquid that burst into flame on contact. Greek Fire (also called wildfire, liquid fire, or Byzantine fire) was a weapon used in the seventh century by the Byzantine Empire (also known as the Greek-speaking Roman Empire).

Greek Fire, supposedly invented by a Syrian Christian refugee named Callinicus, was a totally new type of defensive and offensive weapon. Water did not put it out – in fact, it would even burn underwater. The exact chemical composition of Greek Fire was a closely guarded military secret – so secret, in fact, that the exact formula has been lost to history[2]. We can, however, guess its composition. It probably contained chemicals such as naphtha, niter, sulfur, saltpeter, and phosphorus[3]. It is possible that it contained calcium phosphide, which would ignite upon contact with water.

The exact chemical formula, even though amazing, is not as amazing as the delivery system that the Byzantines used. They managed to shoot this liquid fire from swiveling nozzles mounted on small boats. This was accomplished without thermometers, safety valves, pressure gauges, or other modern technology. Boats drenched with Greek Fire burned – and sailors were unable to quench the fire if it landed on them. Once a soldier or sailor was hit with the liquid, even jumping into the sea would not extinguish the flames. The weapon caused enemies to shiver in terror and capitulate in despair. It has been compared with the modern dread of the atomic bomb.

Sounds like magic, right? That's the key – really successful technology appears like magic. I know this because I was brought up right. I grew up on a diet of truly great science fiction, from truly great authors, and Arthur C. Clarke was one of my favorites. See, I remember Clarke's three laws:

1. When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong[4].
2. The only way of discovering the limits of the possible is to venture a little way past them into the impossible.
3. Any sufficiently advanced technology is indistinguishable from magic[5].

I remember one of my magic moments. In 1981, I was working on my computer science degree at the University of Central Florida. Taking five computer science courses at the same time, I literally lived in the computer center. Large mainframe, limited computer time, crowded cardpunches, etc. Then I found some magic. Ever heard of the Commodore SuperPET, the world's first co-processor computer? It was a standard Commodore PET (with a 6502 processor), plus an expansion board that carried a 6809 processor and 64K more memory. The SuperPET ran versions of APL, FORTRAN, Pascal, and Basic. Although I had to put up my car to get the loan, I plunked down about $5,000 for one (including two 320K disk drives, a dot-matrix printer, and a blazing fast 300 baud modem). By contrast, a mainframe cost hundreds of thousands of dollars and needed a complex supporting infrastructure. My home computer truly was magic! While others waited for hours to punch cards, submit the card deck, and wait for a compile and run, I could sit at home and have UNLIMITED computer time.

Just a few years earlier, the thought of a home computer would have seemed as much like magic or science fiction as *Star Trek* or *Star Wars*. Fact is, it doesn't take long for science fiction magic to become commonplace. CDs. DVDs. HDTV. Hybrid cars. Cell phones. Bluetooth (I don't need a communicator – I just tap my ear, and tell my cell phone who I want to talk to). I wonder how much longer before *Star Trek* and *Star Wars* will look like fact instead of fiction.

Of course, nowadays, we are the ones who make the magic. The Stealth Fighter. The Airborne Laser. The Predator. The Active Denial System. The Space Shuttle. GPS. The Next Generation Destroyer. And almost every other program that the Department of Defense has under development. Science fiction yesterday. Science Fact tomorrow. It's all magic.

Let's hear it for the magic and the engineers who create it.

— David A. Cook, Ph.D.
The AEgis Technologies Group, Inc.
dcook@aegistg.com

Notes

1. See "Greek Fire, Poison Arrows and Scorpion Bombs," by Adrienne Mayor. This book is used throughout this this article. Also see <en.wipipedia.org/wiki/Greek_fire> and <www.greek.org/Romiosini/greek_fire.html>.
2. A legend, however, is that an angel whispered the formula to Emperor Constantine the Great.
3. See "A History of Greek Fire and Gunpowder," by J.R. Partington.
4. Clarke's Law, later the first of the three laws, was proposed by Arthur C. Clarke in the essay *Hazards of Prophecy: The Failure of Imagination*, in "Profiles of the Future." The second law is offered as a simple observation in the same essay; its status as Clarke's Second Law was conferred on it by others.
5. In a revised edition of "Profiles of the Future," Clarke acknowledged the Second Law and proposed the Third in order to round out the numbers, adding "As three laws were good enough for Newton, I have modestly decided to stop there." Out of those three laws, the Third Law is the most known and widely cited <en.wikipedia.org/wiki/Clarke's_three_laws> and <www.commodore.ca/products/pet/commodore_ pet.htm>.

CrossTalk / 517 SMXS/MXDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737